SIMULATED BINARY CROSSOVER FOR REAL-CODED GENETIC ALGORITHMS: DEVELOPMENT AND APPLICATION IN AMBIGUOUS SHAPE MODELLING

by

Ram Bhushan Agrawal

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

APRIL, 1995

Simulated Binary Crossover For Real-coded Genetic Algorithms : Development and Application In Ambiguous Shape Modelling

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

Ram Bhushan Agrawal

to the

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April, 1995

CERTIFICATE

It is to certify that the work contained in this thesis entitled "**Simulated Binary Crossover for Real-coded Genetic Algorithms : Development and Application in Ambiguous Shape Modelling**" by *Ram Bhushan Agrawal*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Kalyanmoy Deb

(Kalyanmoy Deb)
Assistant Professor
Department of Mech. Engg.
I. I. T. Kanpur

April, 1995

Dedicated to
My Dear Diddi
whose inspiration has
brought me to this stage.

" I am like a child, picking small stones at the sea-shore,
while the great ocean of truth is lying before me. "

-- Sir Issac Newton

ABSTRACT

Optimization is an important issue in a design process. There are many traditional methods for optimization, but they have some inherent drawbacks. Genetic algorithms are relatively new search and optimization techniques which are better at handling some of these difficulties. Usually, a *genetic string* is defined through binarization of the design variables called binary coding. The success of binary-coded genetic algorithms (GAs) in problems having discrete search space largely depends on the coding of the variables and on the crossover operator that propagates building-blocks from parent strings to children strings. In solving optimization problems having continuous search space, binary-coded GAs discretize the search space by using a coding of the problem variables in binary strings. However, the coding of real-valued variables in finite-length strings causes a number of difficulties—inability to achieve arbitrary precision in the obtained solution, inherent Hamming cliff problem associated with the binary coding, and processing of Holland's schemata in continuous search space. Although, a number of real-coded GAs have been developed to solve optimization problems having a continuous search space, the search powers of these crossover operators are not adequate. The search power of a crossover operator can be defined in terms of the probability of creating an arbitrary child solution from a given pair of parent solutions. Motivated by the success of binary-coded GAs in discrete search space problems, in this thesis, we develop a real-coded crossover (we called it simulated binary crossover or SBX) operator whose search power is similar to that of the single-point crossover used in binary-coded GAs. Simulation results on a number of real-valued test problems of varying difficulty and dimensionality suggest that the real-coded GAs with the SBX operator are able to perform as good or better than binary-coded GAs with the single-point crossover. SBX is found to be particularly useful in problems having multiple optimal solutions with a narrow global basin and in problems where the lower and upper bounds of the global optimum are not known a priori. Further, the real-coded GAs with SBX are able to successfully find the global optimum solution in a *blocked* function, which causes problems to other real-coded GAs.

In the second part of this work, we demonstrate the use of GAs with SBX on a design application : *ambiguous shape modelling.* Present CAD systems use quantitative models for representing the object geometries, which are not suitable for applications such as conceptual design, shape abstraction, qualitative pattern recognition etc., where one needs to represent not one, but a class of geometries. One novel mechanism for ambiguous shape modelling is based on qualitative mapping on the parameters of the axis of the shape. However, this approach presents some difficulties in extracting reconstructable shapes in a generalized domain, because of the problems such as simultaneous multiple constraint satisfaction, and generation of inconsistent shapes. This can be improved by using a hybrid qualitative model with variable zones, but generating "desirable" shapes remains a problem. In this thesis, a GA based approach is used to handle these difficulties. SBX has been applied on the parameters of a hybrid qualitative system. The use of GAs makes it possible to provide a visual display to the user (inexact visualization) and to arrive at a number of different shapes satisfying some constraints to different degrees (diversity). Moreover, the fitness criterion for the GA can be defined such that infeasible shapes get eliminated and more desirable shapes get preference. Some test shapes generated by this system are presented in the thesis, which demonstrate the viability of the system.

ACKNOWLEDGEMENTS

I feel extremely happy to express my sincere gratitude towards Dr. Kalyanmoy De my thesis supervisor, for introducing me to the wonderful world of genetic algorithn The open atmosphere for exchanging the views with him, coupled with his consta encouragement right from the beginning, was the main driving force throughout tl work. My special thanks goes to Dr. Amitabha Mukerjee, Asst. Prof., Mech. Eng Dept., for providing me the concepts and ideas in the second part of this thes without which this work could never have come to an end. A number of discussio with him has been of considerable help to me in this work.

I am grateful to my senior colleagues, particularly to Mr. P.V.M. Rao and Mr. N. Srii vas, for their technical help from time to time. I also acknowledge the financial su port by the Department of Science and Technology, Govt. of India, under project r SR/SY/E-06/93. Finally, I would like to thank all the learned authors and researche whose research works have been referred in the thesis.

Ram Bhushan Agrawal

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A decision process often involves setting of some decision variables in order to achieve certain objectives within available resources and constraints. For example, a new product may be needed to be designed in such a way so as to minimize its production cost and maximize its marketability; a manufacturing process must be planned so as to minimize the total process time; a satellite should be launched into a path which minimizes total energy consumption. All these are examples of *optimization,* which deals with finding a set of decision variables such that an objective function is maximized (or minimized).

Optimization is a well studied area and various algorithms for optimization are available (Rao, 1991). Genetic Algorithms (GAs) are one of the recent techniques used for optimization and are gaining popularity both in academics and in industry because of certain unique features which are discussed in the next chapter.

Typically, optimization is accompanied by *constraint satisfaction* which relates to searching a solution satisfying some predefined constraints. This is another area where extensive research has been done. Traditionally, optimization and constraint satisfaction are considered to be two independent fields and algorithms for them have been developed separately. However, standard optimization techniques are slow in constraint satisfaction and standard constraint satisfaction techniques are inefficient in optimization. This is one of the serious problems in design process. It is interesting to note that GAs can be used for 'constraint satisfaction' and 'optimization' both. Hence, in this thesis, we focus on GAs as a search and optimization tool. To motivate this discussion, we provide an example.

Let us consider a specific task, that of designing a knife cross-section. We begin with a class of possible shapes for cross section (Fig. 1.1). Note the progressive refinement done by designer on the basic shape. In the first case, he finds the cutting edge is very thin, leading to high wear rates. So he refines the cutting edge in the next design. There may be other constraints that need to be "satisfied", e.g. aspect ratio of the shape, which might be needed to fall within some range. When the designer has roughly decided the shape, he may wish to 'optimize' the shape attributes, e.g. cutting angle and bevel thickness. This example depicts that a design process normally involves optimization as well as constraint satisfaction. A GA based approach can be used to handle both of these aspects.

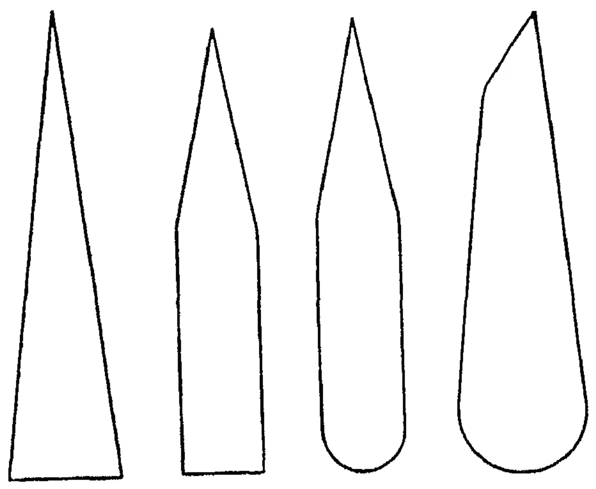

Figure 1.1: Designing the cross-section of a knife. Note the basic similarity in the shapes.

We begin the thesis in chapter 2 by an introduction to GAs. That chapter also addresses the problems associated with the present approach of GA (specifically, the binary GA) when applied to continuous search domain. To combat these problems, some researchers advocate use of real-coded GAs (Davis,1991a; Wright, 1991; Radcliffe, 1993; Eshelman and schaffer, 1993). A real-coded GA is different than standard GA in the sense that it uses floating-point numbers to represent the variables in contrast to binary strings used by standard GA. Since crossover operator is the main search operator in working of GAs, the main emphasis given by different researchers is on this operator only. A number of real-coded GA crossovers have been reported in the literature (Wright, 1991; Eshelman and Schaffer, 1993), but the search powers of these operators are not found to be adequate in a wide variety of problems. In this thesis, we have devised a new crossover operator for real-coded GA termed as simulated binary crossover (SBX), which has much more flexibility in terms of search power. The most striking feature of this crossover is that it simulates the effect of standard binary crossover in a real-coded variables' domain.

Chapter 3 presents the results of GAs with SBX operator on a number of test problems. Most of these test problems are chosen from the literature, and cover a variety of functions — from one-variable to multivariable, unimodal to multimodal, continuous to discrete, and smooth to noisy. The obtained results are encouraging and show that SBX has performed either better than or similar to binary crossover operator.

Gaining the confidence in the performance of SBX on several test problems, we then apply SBX to *ambiguous shape modelling* in chapter 4. Even though this has been taken only as an application problem for real-coded GA, it is important in its own. Note that even though the shapes in fig. 1.1 are different from each other, there is some basic similarity in all the shapes (length is significantly more than the width, and width reduces from one end to other). This type of shape abstraction is what a designer will like to keep in his model. Almost all the present CAD models are quantitative in nature, storing exact information about the shape. However, such models are of little use at conceptual stage of design, when the designer hasn't fixed up the shape, but still wants to try various options. For this, a *qualitative model* of the shape is required, which should be able to store not just a single shape, but a class of shapes. The model developed here uses a novel mechanism for representing the class of shapes which is based on 'smudging' the axis of a shape (also known as *medial axis transform or MAT*) proposed by King and Mukerjee (1990). One problem with this approach is that this mechanism sometimes generate inconsistent shapes. In this thesis, we use the same mechanism, coupled with GAs to eliminate inconsistent shapes and to find shapes with more desirable attributes. For this purpose, various penalty functions have been defined relating to properties of shapes generated. GA with SBX is then applied to eliminate inconsistent shapes and generate desirable shapes. Use of GA makes it possible to search non-linearly in a highly complex domain of search space. for the solutions which will be otherwise almost impossible to obtain by traditional methods. The test results obtained present a confidence about viability of the system for ambiguous shape modelling

A discussion on various possible avenues of further research on the current topic is outlined in chapter 5. The final chapter discusses a summary of the work performed in this thesis.

Chapter 2

Genetic Algorithms : Development of SBX

Optimization is an important issue in a decision process. In its most general sense, 'optimization' means maximizing (or minimizing) an objective by setting appropriate values for its decision variables. Most of the standard optimization algorithms start with a inital guess point, and then the solution gets updated iteratively depending upon some transition rules (Rao,1991). These algorithms have been extensively used in real-world applications. However, there are mainly two drawbacks associated with the traditional optimization techniques. Firstly, they are not *robust*, in the sense that a particular algorithm suits only for a limited class of optimization problems. Thus, the user needs to know a number of algorithms and their respective domain of applicability. It does not harm to know about a number of optimization algorithms, but it is definitely better to know one robust algorithm applicable to many problems. Secondly, most of the traditional optimization techniques are *local-search* algorithms. Consider, for example, the function shown in fig. 2.1. If the starting point is chosen from the range AB, the algorithm is likely to converge at point L, which is a *local* optimum. The *global* optimum G can only be achieved if we start with a point in the range BC. In real-world situations, the objective functions are much more complex both in dimensionality and behaviour, and it may be almost impossible to select a suitable starting point to reach to the global solution. In the next section, we look at a very different type of optimization technique — genetic algorithm — which is somewhat more efficient in handling the above difficulties.

2.1 Genetic algorithms

Genetic algorithms (GAs) are nontraditional search and optimization techniques based on an analogy with natural genetics (Goldberg, 1989). They have been applied successfully into a

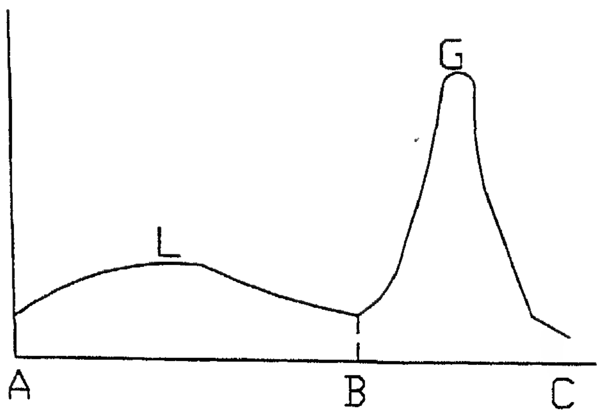

Figure 2.1: A simple bimodal function. Points L and G denote local and global optima. respectively.

wide variety of problems. and have yielded results which were either not possible or were very difficult to achieve using traditional methods. There are some salient features of GAs which put them into an altogether different class from traditional methods. These are as follows :

1. GAs have *global perspective*, i.e., they don't easily get stuck in local optima.
2. GAs are *robust* techniques, i.e. they can be applied to many classes of optimization problems.
3. GAs use a *population* of solution points, and not a single point.
4. GAs use *probabilistic* rules for refinement of solution points and not the *deterministic* rules.
5. GAs don't require gradient or any other auxiliary information about the problem.
6. GAs usually work with a coding of variables and not the variables themselves[1].

A full description of working of GAs can be found in (Goldberg, 1989). However, we present here a brief overview of it's working mechanism for the sake of continuity.

2.1.1 Working of simple genetic algorithm

To explain the working of GAs, we first consider a one-variable unconstrained function :

$$\text{Maximize } f(x) \qquad \text{for} \quad x_{min} \leq x \leq x_{max}$$

[1]However, this feature of GA is not absolutely necessary In a later section and in most part of this thesis we show how the coding can be avoided.

An extension for multi-variable problems will be discussed later. In order to solve the above problem using GAs, the variable x is typically coded in some string structures. Binary coding is normally used and the decoded value of string is linearly mapped on the variable range The length of the string is usually determined according to the desired precision level of the solution. Thus the precision level acquired by using ℓ-bit long string will be $(x_{max} - x_{min})/2^{\ell}$ With a known coding, any string can be decoded to an x value, which can then be used to find the objective function value. A string's objective function value ($f(x)$) is termed as string's *fitness* in GA terminology.

```
initialize();
evaluation();
while (termination criteria are unsatisfied)
{
   reproduction();
   crossover();
   mutation();
   evaluation();
}
```

Figure 2.2: A pseudo code for simple genetic algorithm.

The working of GAs can be broadly illustrated by fig. 2.2. First, a population of randomly-generated strings is created. The number of strings in a population is called *population size* and usually supplied by the user. For more complex problems, larger population sizes are normally used. Once the population is initialized, each string in the population is evaluated and assigned it's *fitness* value, which is a measure of it's *goodness* For a maximization problem, usually the objective function value itself is treated as string's fitness value. After this, the three main operators — reproduction, crossover and mutation — are applied on the population and it is again evaluated. This is then tested for termination criteria. If they are not met, the population is modified again using the same operators and then evaluated. This process is continued till termination criteria are satisfied. We now take a closer look at the three operators :

- *Reproduction :* This operator selects better strings from the population and sends copies of these strings in a temporary population called *mating pool.* This is analogous to Darwin's *survival of the fittest* principle. Usually a probabilistic approach is used to select the strings, which results in selection of bad strings once in a while (although with a small probability). At first sight, it sounds awkward, but a careful thinking will convince that this is necessary if we want GA not to get stuck in local optima. There are a number of reproduction schemes which can be used for this purpose, e.g. proportionate selection, tournament selection, and stochastic remainder selection (Goldberg and Deb, 1991).

- *Crossover :* The reproduction operator can't generate new solution points. This job is taken care by crossover. Crossover operator is analogous to recombination of genes of parents in biological systems. According to Deb (1993) :

  > ...the purpose of the crossover operator is two-fold. The main purpose of the crossover operator is to search the parameter space. Other aspect is that the search needs to be performed in a way that the information stored in the parent strings are maximally preserved, because these parent strings are instances of good strings selected using the reproduction operator.

  In one of the various possible crossovers (Spears and De Jong, 1991), single point crossover, a random cross-site is chosen along the length of string, and the bits on either sides of cross-site are swapped between the parent strings. In section 2.2, we will be dealing with this crossover in detail.

- *Mutation :* It is considered to be a secondary operator in GAs, and its main function is to guard the population against premature convergence. Mutation flips a randomly chosen bit of a string from 1 to 0 (or vice-versa). Usually, mutation is applied with a very small mutation probability (p_m), so that only very few strings undergo this flipping. To understand the necessity of mutation, consider a particular case in which the last bit of the string at global solution is 1. Now, suppose that all the strings in a generation have a 0 in that place, then neither reproduction nor crossover can generate the global solution. But introduction of mutation makes it possible (though not guaranteed) to flip the last bit from 0 to 1.

For a multivariable problem, all strings corresponding to each variable are concatenated to form a bigger string and then this string is used for GA search. It looks surprising, but it is true that the three random and simple-looking operators described above constitute a highly nonlinear and robust search and optimization paradigm. In the next section, we look at some intutive reasoning behind the working of GAs.

2.1.2 Similarity templates (schemata)

The main secret of success of binary GA is the processing of *schemata* (plural of *schema*) which goes on in parallel with the operations of GAs. A schema is a similarity template (Holland, 1975) and is represented by combination of three symbols — 1, 0 and $*$ (don't care). In general a schema represents a class of strings. For instance, a schema (1 $*$ $*$ $*$ 0) means all those 5-bit strings which have a 1 in the first place and a 0 in final place. In the variable space, a schema represents a region which may be or may not be continuous. For example, schema (1 $*$ $*$ $*$ $*$) represents the right half of the search space, whereas schema ($*$ $*$ $*$ $*$ 1) represents every alternate string. It has been shown elsewhere (Goldberg, 1989) that those schemata, which are good, will be getting exponentially higher number of copies in forthcoming generations and those which are poor will slowly die out. This phenomenon, of schemata being processed in parallel is known as *implicit parallelism* in GAs and is one of the important underlying theories of GAs.

2.1.3 Shortcomings of binary GA

Binary GA has been popularly used to solve both types of problems having discrete or continuous variables. The success of binary-coded GAs in problems having discrete search space is largely due to the careful choice of a coding and crossover operator that process useful schemata efficiently and propogate it in successive generations. However there are certain shortcomings associated with the binary coding, especially when applied to continuous domain. These can be summarized as below :

1. *Artificial hinderance to search created by Hamming cliff*

   Hamming cliff (Davis, 1991b) refers to sudden changes in genic space with a gradual change in variable space. For instance, even though the two strings 011111 and 100000 are separated by only one unit in variable space, transition from one string to other in genic space requires change in all the bits. Thus if the global solution is at 100000 and the population converges to 011111, it will be very difficult to reach the solution even by using mutation. Some researchers advocate use of *gray coding* (Schaffer et.al., 1989) to avoid this problem, but that is merely a substitution of one problem for another, because use of gray coding introduces artificial non-linearity in the problem.

2. *Fixed precision level*

   The precision acquired in the solution is fixed in binary GA because it depends on the string length specified which should be decided before-hand. The more is the required precision, more should be string length. For larger string lengths, the population size requirement is large (Goldberg, Deb and Clark. 1992), thereby increasing the computational overheads. Even with a reasonable string length, there is always the danger that one has not allowed enough precision to represent parameter values that produce the best solution values. Further, a careful thinking reveals that binary GA gives equal importance to all the regions in search space as far as precision is concerned, whereas in an ideal algorithm, the precision level should be dependent on fitness of regions, i.e. more precision for better regions, and less precision for bad ones.

3. *Fixed variable bounds*

   In a binary-GA, the bounds of every design variables are fixed before-hand, to facilitate mapping from binary coding to variable-space. In certain real-world applications, the user may not be knowing these bounds. If a narrow bound is specified, there is always a danger that the global optimum is not enclosed in the specified bounds. On the other hand, a too-wide range of variables require a larger string length (if precision level is to be kept the same) which increases the risk of premature convergence. In an ideal algorithm, the search should be extended to practically anywhere for an unconstrained optimization problem.

4. *Processing of unimportant schemata*

   There are a large numbers of Holland's schemata that are processed during a GA search. If we think about this processing a bit closely, we see that not all schemata that are processed are important. In a continuous search space, the meaningful schemata are those which represent continuous regions of the search space. For example, the schema (1 1 * * *) represents the rightmost quarter of the search space, so it may be meaningful. But the schema (* * * * 1) represents every alternate point in search space. This may not be of any importance in continuous domains. Hence, the crossover needs to be redesigned in order to process more meaningful schemata relevant to continuous search space.

Realizing the above aspects, a growing number of researchers have started paying attention to *real-coded GAs* which use parameter values themselves instead of any coding. Obviously, this type of GA requires crossover and mutation operators which are radically different from their counterparts in binary GA. In section 2.3 we give a brief review of existing crossovers for real-coded GAs and section 2.4 deals with a new type of real-coded crossover, SBX, developed in this thesis. The real-coded GA using SBX is free from the shortcomings of binary GA discussed above. At the same time, SBX simulates the beneficial aspects of binary crossover. But before we discuss that, we need to look at search power of binary crossover, which is discussed in next section.

2.2 Binary crossover operator

In many applications of binary-coded GAs, a single-point crossover is used. In a single-point crossover, a random cross site along the length of the string is chosen and the bits on one side of the cross site are swapped between two parent strings (Fig 2.3). In a single-variable optimization problem, the action of the crossover is to create two new children strings from two parent strings. In a multi-variable optimization problem, each variable is usually coded in a certain number of bits (collectively called a substring) and these substrings are combined to form a bigger string. Since, in a single-point crossover operator, only one cross site is chosen from the complete string, the action of the single-point crossover may disrupt one variable or a combination of continuous variables simultaneously. In the following, we calculate the probability of creating a child point from two chosen points under the single-point crossover.

2.2.1 Search power of single-point crossover

In this thesis, we assume that the search power of a crossover operator is a measure of how flexible the operator is to create an arbitrary point in the search space. Here, we calculate the search power of a crossover operator by finding the probability of creating an arbitrary point in the search space from two given parent points[2]. To make the analysis simpler, we also assume that the function is a single-variable function.

When two parent binary strings of length ℓ are crossed at an arbitrary site, the contents on either side of the cross site get swapped between the parents. Let us imagine that the cross site is k bits from the right of the string. We also assume that the leftmost bit is the highest significant bit in the underlying binary coding. If the decoded value of the rightmost k bits in the first parent is A_1 and the decoded value of the rest $(\ell - k)$ bits is B_1, then the decoded

[2]The analysis in this section also appears in our earlier paper (Deb and Agrawal, 1994).

$$\begin{array}{cccc|cccc c}
\multicolumn{4}{c|}{B_1} & \multicolumn{3}{c}{A_1} & & \text{DV}\\
0&1&1&1&0&0&1&&57\\
1&1&0&0&1&0&1&&\underline{101}\\
\multicolumn{4}{c|}{B_2}&\multicolumn{3}{c}{A_2}&\text{Avg.}&79
\end{array}
\Longrightarrow
\begin{array}{cccc|cccc c}
\multicolumn{4}{c|}{B_1} & \multicolumn{3}{c}{A_2} & & \text{DV}\\
0&1&1&1&1&0&1&&61\\
1&1&0&0&0&0&1&&\underline{97}\\
\multicolumn{4}{c|}{B_2}&\multicolumn{3}{c}{A_1}&\text{Avg.}&79
\end{array}$$

Figure 2.3: The action of single-point crossover on two random strings is shown. DV stands for the decoded parameter value. Notice that the average of the decoded parameter values is the same before and after the crossover operation.

value of the string is as follows:

$$x_1 = B_1 2^k + A_1. \tag{2.1}$$

The decoded value of the second parent string can also be calculated by knowing A_2 and B_2 as the decoded values of the right k bits and the rest $(\ell - k)$ bits of the string, respectively: $x_2 = B_2 2^k + A_2$. Figure 2.3 shows the single-point crossover operation on two arbitrary strings and the resultant children strings obtained by crossing at a particular site. The effect of the single-point crossover operator is to swap the values A_1 and A_2 between the two parent points. Thus, the decoded values of the children strings are as follows:

$$\begin{aligned} y_1 &= B_1 2^k + A_2, \qquad (2.2)\\ y_2 &= B_2 2^k + A_1. \qquad (2.3)\end{aligned}$$

The above equations bring out an important property of the single-point crossover. Notice that the mean of the decoded values of the parent strings ($(x_1 + x_2)/2$) is the same as that of the children strings ($(y_1 + y_2)/2$). This can also be observed in Figure 2.3, by calculating the decoded parameter values of both parent and children strings. It is interesting to note that this property of decoded parameter values holds good for real parameter values of parent and children points for a linear mapping rule. Assuming a linear mapping rule, the parent points can be obtained as follows:

$$p_i = m x_i + t, \qquad i = 1, 2, \tag{2.4}$$

where m and t are two constants. The children points can also be computed using the same mapping rule as follows:

$$c_i = m y_i + t, \qquad i = 1, 2. \tag{2.5}$$

Using the above two equations and the previous observation on y_i, it is trivial to show that mean of the parent points is the same as that of the children points. In other words, the children points are equidistant from the mid point of the parent points. We shall use this property to develop the real-coded crossover operator later. Further, we observe that the children points may lie inside or outside the region bounded by the parent strings depending on the the strings and the location of the cross site. In order to define the spread of the children points with respect to that of the parent strings, we define a *spread factor* β as the ratio of the spread of children points to that of the parent points:

$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right| . \tag{2.6}$$

With the above definition of the spread factor, we classify crossovers in three different classes:

Definition 1 *All crossover operations having a spread factor $\beta < 1$ are contracting crossovers.*

When the children points are enclosed by the parent points, the spread of the children points (the absolute difference in c_1 and c_2 values) is smaller than that in parent points. Thus, the spread factor β is less than one. Since the children points usually replace the parent points in nonoverlapping genetic algorithms, this crossover seems to have a *contracting* effect on the parent points.

Definition 2 *All crossover operations having a spread factor $\beta > 1$ are expanding crossovers.*

When the children points enclose the parent points, the absolute difference in children points is more than that of the parent points. This crossover has an effect of *expanding* the parent points to form the children points.

Definition 3 *All crossover operations having a spread factor $\beta = 1$ are stationary crossovers.*

In stationary crossovers, the children points are the same as the parent points. One interesting feature of the spread factor β is that it *hides* the effect of the true spread of the children points. The spread factor β signifies a spread of children points relative to that of the parent points. For a fixed value of β, two close children points can be obtained if the parent points are close to each other or two distant children points could be obtained if the parent points are far away from each other. In a single-point crossover, the occurrence of a contracting, an expanding, or a stationary crossover depends on the parent strings and the location of the cross site.

With the above definitions of different types of crossovers, let us now compute the probability of occurrence of three different types of crossovers for any two random parent strings. We first consider that two ℓ-bit parent strings are crossed at a site $k \in (0, \ell - 1)$ from the rightmost bit. Once again, we assume that the leftmost bit is the most significant bit. Representing two parent strings as $a_i, b_i \in \{0, 1\}$, we write the decoded values as follows:

$$x_1 = \sum_{i=0}^{\ell-1} a_i 2^i, \tag{2.7}$$

$$x_2 = \sum_{i=0}^{\ell-1} b_i 2^i. \tag{2.8}$$

Using the property of single-point crossover operator given in equations 2.3 and 2.3, we obtain the decoded values of the children strings as follows:

$$y_1 = \sum_{i=0}^{k-1} b_i 2^i + \sum_{i=k}^{\ell-1} a_i 2^i, \tag{2.9}$$

$$y_2 = \sum_{i=0}^{k-1} a_i 2^i + \sum_{i=k}^{\ell-1} b_i 2^i. \tag{2.10}$$

For a linear mapping rule (equations 2.4 and 2.5), the spread factor defined in equation 2.6 can be written in terms of decoded values of parent and children strings. Thus, equations 2.8 through 2.10 can be used to write the spread factor β. But we simplify the expression for the spread factor by assuming a parameter $u_i = (a_i - b_i)$ for all $i = 0, 1, \ldots, (\ell - 1)$. Note that each u_i can take a value -1, 0, and 1.

$$\begin{aligned}
\beta(k) &= \left| \frac{y_1 - y_2}{x_1 - x_2} \right|, \\
&= \frac{\sum_{i=k}^{\ell-1} u_i 2^i - \sum_{i=0}^{k-1} u_i 2^i}{\sum_{i=k}^{\ell-1} u_i 2^i + \sum_{i=0}^{k-1} u_i 2^i}, \\
&= \frac{1 - \zeta(k)}{1 + \zeta(k)},
\end{aligned}$$

where $\zeta(k) = \sum_{i=0}^{k-1} u_i 2^i / \sum_{i=k}^{\ell-1} u_i 2^i$. The above term for β is valid for any two parent strings (or in other words for any u_i values) and for any crossover site (or for any k). But, let us first consider the contracting crossovers (where $0 \leq \beta \leq 1$). In order to simplify our analysis further, we assume that parent strings are two extreme strings having all ones (111...1) and all zeros (000...0). In this case, the difference in allele values in each bit is one (or $u_i = 1$ for $i = 0, 1, \ldots, (\ell - 1)$). Substituting $u_i = 1$ in the above expression, we obtain the following expression for β:

$$\beta(k) = 1 - 2(2^k - 1)/(2^\ell - 1) \approx 1 - 2^{(-\ell+1)} 2^k. \tag{2.11}$$

The above spread factor for two extreme strings of length 20 crossed at different sites is plotted in figure 2.4. It is clear from the figure that for most cross sites k, the spread factor $\beta \approx 1$.

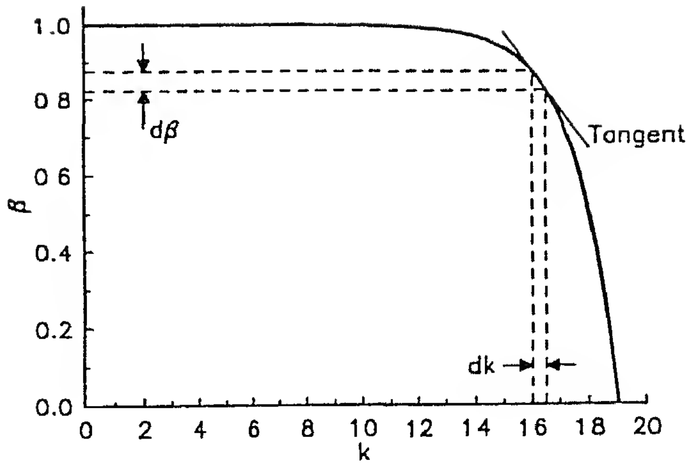

Figure 2.4: The distribution of $\beta(k)$ versus k for $\ell = 20$ is shown.

With a distribution of the spread factor as a function of the cross site, we can now compute the probability of creating any arbitrary string (a representative β) from two given parent strings. Let us consider figure 2.4 again. For a particular value of β, we consider a small range $d\beta$ and find how many crossover operations (dk) would produce children points in that range. This can be achieved by calculating the slope of the above expression at the corresponding β:

$$dk = \frac{1}{\left|\frac{d\beta}{dk}\right|_{\beta}} d\beta. \tag{2.12}$$

Denoting the probability of occurrence of β by $C(\beta)$ and knowing that β lies between zero and one for contracting crossovers, we obtain the distribution for $d\beta \rightarrow 0$:

$$C(\beta) = \frac{1/\left|\frac{d\beta}{dk}\right|_{\beta}}{\int_0^1 \left(1/\left|\frac{d\beta}{dk}\right|_{\beta}\right) d\beta}. \tag{2.13}$$

The denominator of the right-side expression is a constant quantity. Here, we are not interested in computing the above distribution exactly, rather we are interested in finding the functional relationship of the distribution $C(\beta)$ with β. Thus, it will suffice here to compute the numerator of the above expression only. By differentiating equation 2.11 with respect to k and substituting the expression for k in terms of β, we obtain the numerator of the right-side expression in equation 2.13 as follows:

$$C(\beta) = \frac{\kappa}{1-\beta}, \tag{2.14}$$

where κ is a constant term. The functional form of the above distribution is shown in figure 2.5 (for $0 \leq \beta \leq 1$). The figure shows that as β is increased towards one, the probability increases.

Thus, the probability of creating children points closer to the parent points is larger than that of points far away from the parents.

The distribution for expanding crossovers ($\beta > 1$) can be obtained by considering two other parent strings[3] and by using the above procedure. However, this distribution can also be obtained from that of the contracting crossovers by realizing a simple probabilistic event. If two children strings found in a contracting crossover are crossed at the same site, the same parent strings are obtained. Thus, for each contracting crossover with a spread factor β, there exists an expanding crossover with a spread factor $1/\beta$. In other words, we can argue that the probability of occurrence of contracting crossovers having a spread factor between β and $(\beta + d\beta)$ would be the same as the probability of occurrence of expanding crossovers with a spread factor between $1/(\beta + d\beta)$ and $1/\beta$. Summing these probabilities for all contracting crossovers in the range $0 \leq \beta \leq 1$, we conclude that the overall probability of contracting crossovers is the same as that of the expanding crossovers[4]. Equating the probability of expanding crossovers ($\mathcal{E}(\beta)$) having spread factor between β and $(\beta + d\beta)$ and that of contracting crossovers ($\mathcal{C}(\beta)$) having spread factor between $1/(\beta + d\beta)$ and $1/\beta$, we obtain the following:

$$\mathcal{E}(\beta)\left[(\beta + d\beta) - \beta\right] = \mathcal{C}(\frac{1}{\beta})\left[\frac{1}{\beta} - \frac{1}{\beta + d\beta}\right]. \tag{2.15}$$

Rearranging terms and in the limit $d\beta \rightarrow 0$, we obtain

$$\mathcal{E}(\beta) = \frac{1}{\beta^2}\mathcal{C}(\frac{1}{\beta}). \tag{2.16}$$

With this relationship between the probability of occurrence of contracting and expanding crossovers, we only have to know the probability distribution for only one type of crossovers. Since their sum is the overall probability of all crossovers, the overall probability for either contracting or expanding crossovers must be equal to 0.5. For the probability distribution given in equation 2.14, we obtain a probability distribution for expanding crossover using the above equation:

$$\mathcal{E}(\beta) = \frac{\kappa}{\beta(\beta - 1)}. \tag{2.17}$$

This distribution is plotted in the region $\beta \geq 1$ in figure 2.5. The figure shows that the probability of creating children strings having a large β is small.

[3]Notice that with two extreme strings considered in the contracting crossovers the expanding crossover is not possible.

[4]Of course, the stationary crossovers can be considered as either contracting or expanding. For the sake of argument, we can assume that half of the stationary crossovers are contracting and rest half of the stationary crossovers are expanding.

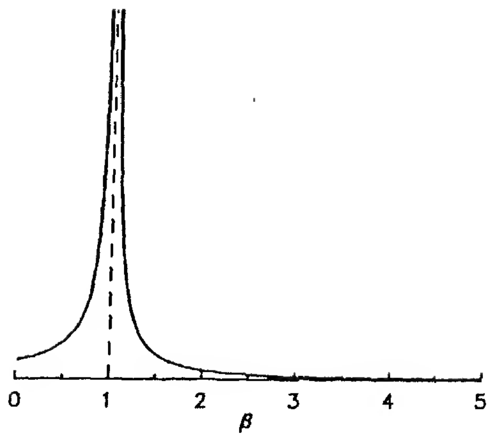

Figure 2.5: Probability distributions of contracting and expanding crossovers on two extreme binary strings are shown.

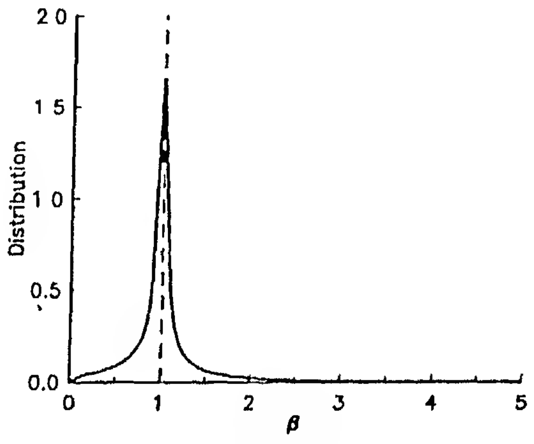

Figure 2.6: Probability distributions of contracting and expanding crossovers on all pairs of binary strings of length 15 are shown.

Recall that equations 2.14 and 2.17 are derived only for two extreme binary strings. In order to investigate the above distributions and probability distributions for other parent pairs, we create all pairs of binary strings of a particular length and cross them at all possible cross sites. For each crossover, the corresponding spread factor β is calculated and arranged in ascending order. The experiment is continued for string lengths of 10, 15, and 20. Figure 2.6 shows the distribution for $\ell = 15$. Distributions obtained for other string lengths also look similar. The fig. 2.6 shows that the probability of occurrence of $\beta \approx 1$ is more likely than any other β value. If the parents are closer, the spread of the two likely children is also smaller. On the other hand, if the spread of parents is more, the spread of likely children is also large. This aspect of the crossover operator distinguishes it from the mutation operator. In a mutation operator, the probability of a mutated string close to the parent string is higher, but this probability is usually constant and depends only on one string.

The next section deals with the existing real-coded crossover operators.

2.3 Existing real-coded crossover operators

Even though the shortcomings of binary GAs can be removed by real-coded GAs, the interest in the study of real-coded GAs has begun very recently, and there still remain some objections to real-coded GAs which must be resolved. Most of the theoretical objections to real-coded GAs assume that the crossover operator operates at parameter boundaries. But many implementors of real-coded GAs use crossover operators with extensive search power. The real-coded crossovers present in literature are as below :

1. *Davis's crossover.* Davis (1991a) has devised a crossover that averages some of the variables. Thus if two parent points are p_1 and p_2, the child point is taken as $0.5(p_1+p_2)$.

2. *Wright's linear crossover.* Wright (1991) suggested using a linear crossover operator which generates three child points $0.5(p_1+p_2)$, $(1.5p_1-0.5p_2)$, and $(-0.5p_1+1.5p_2)$. This study also suggested a crossover operator that creates a point by randomly perturbing a single variable in its range.[5]

3. *Radcliffe's flat crossover.* This crossover (Radcliffe, 1993) chooses values for a child point by uniformly randomly picking a value from the range p_1 to p_2. Thus it uses a probabilistic approach to create a child point, which is desirable in GA.

4. *Eshelman's blend crossover (BLX-α).* Eshelman and Schaffer (1993) have suggested a blend crossover (BLX-α) operator. For two parent points p_1 and p_2 (assuming $p_1 < p_2$), the BLX-α randomly picks a point in the range $(p_1 - \alpha(p_2 - p_1),\ p_2 + \alpha(p_2 - p_1))$. If α is zero, this crossover becomes same as Radcliffe's flat crossover. In a number of test problems, they have reported that BLX-0.5 (with $\alpha = 0.5$) performs better than the BLX operators with any other α value.

2.3.1 Search power of real-coded crossovers

For the above real-coded crossovers, we investigate the search power, as defined in the previous section. The search powers of Davis's crossover or Wright's linear crossover are clearly not adequate, because only one and three children points can be obtained from two parent strings,

[5]However, this is more like a mutation operator, rather than a crossover.

respectively. Radcliffe's flat crossover or Eshelman and Schaffer's BLX-0.0 operator has the flexibility of creating any point enclosed by the parent points, thus having a better search power. Eshelman and Schaffer (1993) have also introduced concept of *interval* schemata, and claimed that BLX crossover preserves the interval schemata representing continuous regions in the search space.

Interval schemata are defined by all possible subranges in the variable space. Considering a variable whose range is (0,k), there will be a total of $0.5(k+1)(k+2)$ interval schemata according to the formulations of Eshelman and Schaffer. It is assumed that the points represented by an interval schemata are bounded by integers only. We consider any two parent points in the variable range, and then calculate the number of common schemata between them and an arbitrary children point. When these are plotted in terms of spread factor β, we note that they decrease linearly for $\beta > 1$, as shown in the fig. 2.7. A detailed analysis on this issue can be found in our earlier paper (Deb and Agrawal, 1994). It is to be noted that BLX-0.5 deviates from this ideal distribution, in the sense that it gives equal probability to all β values in the range (0,1.5). Hence, we see that the search power of BLX-0.5 is radically different than binary crossover. It may be argued that BLX-0.5 has higher schemata destruction rates as compared to binary crossover. Motivated by this observation, we devise a new crossover (we call it simulated binary crossover) in the next section.

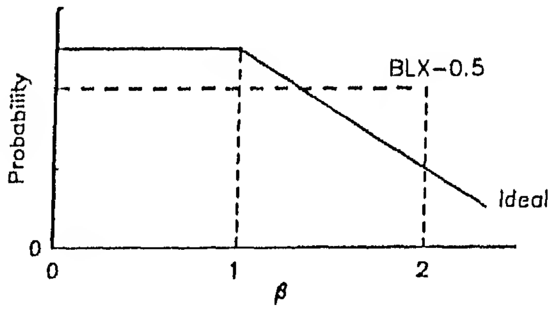

Figure 2.7: Probability distributions of BLX-0.5 and the ideal distribution are shown with spread factor β.

2.4 Simulated binary crossover (SBX)

Simulated binary crossover developed in the present work is a different type of real-coded crossover in a sense that it uses a particular probability distribution to generate children points. We have designed this probability distribution to be similar to that found in actual

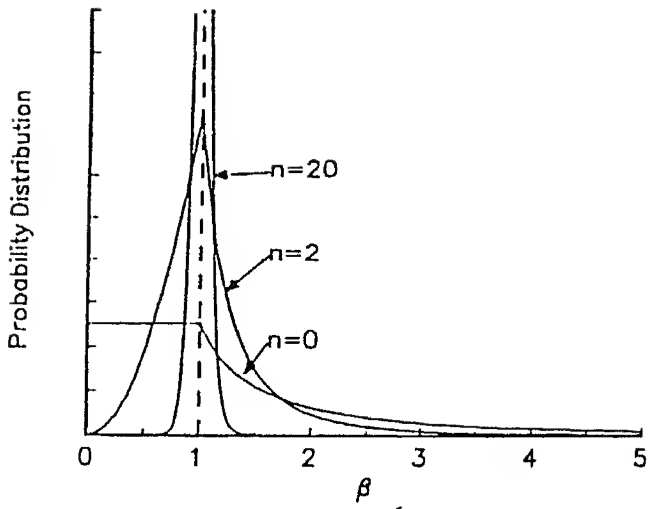

Figure 2.8: Probability distributions of contracting and expanding crossovers for SBX model are shown.

binary crossover. One property of a probability distribution is that the cumulative probability of all possible states is equal to one. But the integration of the probability distribution in equation 2.14 over all values of β is not a finite number. this is because at $\beta = 1$ the probability does not exist. Hence. it is not a valid probability distribution. Moreover. recall that the distribution obtained in equation 2.14 is only for two extreme binary strings as parents. For contracting crossovers, we assume a simple polynomial probability distribution which is similar to the distribution in equation 2.13 and to that shown in figure 2.6.

$$C(\beta) = 0.5(n+1)\beta^n. \tag{2.18}$$

where n is any nonnegative real number. The factor 0.5 makes the cumulative probability at $\beta = 1$ equals to 0.5. For expanding crossovers. the probability distribution can be derived by using equation 2.16 and 2.18:

$$\mathcal{E}(\beta) = 0.5(n+1)\frac{1}{\beta^{n+2}}. \tag{2.19}$$

A large value of n allows a better approximation of the above distribution with equation 2.14 for large values of β. A small value of n makes a more uniform distribution in the range $0 \leq \beta \leq 1$. A value of $n = 0$ makes a perfectly uniform distribution in that range (Figure 2.8). A moderate value of n (2 to 5) matches closely with the simulation results for the single-point crossover in binary-coded GAs. The probability distribution for contracting and expanding crossovers for various values of n are shown in figure 2.8.

Using the probability distribution given by equations 2.18 and 2.19. we now present the

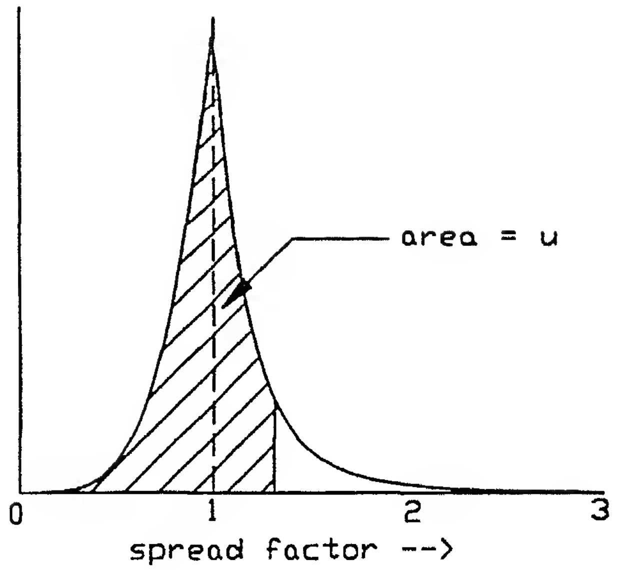

Figure 2.9: Calculation of the spread factor β from a given random number u.

algorithm for calculating children points with the SBX crossover operator.

SBX Algorithm

INPUT : p_1, p_2 (Parent points)

OUTPUT : c_1, c_2 (Children points)

- Generate a random number u such that $0 \leq u \leq 1$
- Calculate the spread factor,

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{(n+1)}} & \text{if } u \leq 0.5, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{(n+1)}} & \text{otherwise.} \end{cases}$$

- Calculate children points,

$$c_1 = 0.5(p_1 + p_2) - 0.5\beta(u)|p_2 - p_1|$$

$$c_2 = 0.5(p_1 + p_2) + 0.5\beta(u)|p_2 - p_1|$$

It is interesting to note that the number u used in the above algorithm represents the area under curve from $\beta = 0$ to $\beta = \beta(u)$ (Fig. 2.9). We could have also assumed u to represent the area under curve for $\beta = \beta(u)$ to $\beta = \infty$. In that case, equations for calculating $\beta(u)$ can be suitably derived.

The polynomial probability distribution given in equation 2.18 is one of many possible probability distributions for contracting crossovers. For a different distribution, the corresponding probability distribution for expanding crossovers can be calculated by using equation 2.16. It is worth mentioning here that a distribution for expanding crossover can also be assumed first and then the corresponding probability distribution for contracting crossovers can be computed by replacing $\mathcal{E}(\beta)$ with $\mathcal{C}(\beta)$ in equation 2.16. In all such probability distributions, the probability of creating children points close to the parent points must be more than that of creating children points far away from the parent points (This simulates the effect of the single-point crossover as found in figure 2.4). However, in the remainder of the thesis, we use the polynomial probability distribution given in equations 2.18 and 2.19.

The concept of probability of crossover (p_c) used in canonical GAs still applies to this real-coded GA. This probability means that on an average p_cN of the points in the mating pool are participated in the crossover.

2.4.1 Properties of SBX

The exponent n in the probability distribution equations is an important parameter. It is clear from figure 2.8 that the distribution largely depends on the exponent n. For small values of n, points far away from the parents are likely to be chosen, whereas for large values of n, only points closer to the parents are likely to be chosen. In a sense, the exponent n is similar to the reciprocal of the temperature parameter (T) used in simulated annealing algorithm (Kirkpatrick, et.al., 1983). Ideally, a good search algorithm must have a broad search (with large T or small n) in early generations and as the generations proceed the search must be focussed on to a narrow region (with small T or large n) to obtain better precision in the solution. However, for brevity, we use a fixed compromised n for the entire simulation.

Let us now check the validity of equations 2.18 and 2.19 as a probability distribution curve. Any ideal probability distribution curve should be at least C^0 continuous and the area under curve should be equal to unity.

Area under curve for $\beta = 0$ to 1,

$$\begin{aligned} A_1 &= \int_{\beta=0}^{1} 0.5(n+1)\beta^n d\beta \\ &= 0.5(n+1)\left[\frac{\beta^{n+1}}{(n+1)}\right]_0^1 \\ &= 0.5 \end{aligned}$$

Area under curve for $\beta = 1$ to ∞,

$$A_2 = \int_{\beta=1}^{\infty} 0.5(n+1)\frac{d\beta}{\beta^{n+2}}$$

$$\begin{aligned} &= 0.5(n+1)\left[-\frac{1}{(n+1)\beta^{n+1}}\right]_1^\infty \\ &= 0.5 \end{aligned}$$

Hence areas under curve for contracting and expanding crossovers are separately equal to 0.5 and when taken together, the area under whole curve sums to unity. Also note that the ordinate of distribution curve for $\beta = 1$ is found to be $0.5(n+1)$ by both equations 2.18 and 2.19 which ensures C^0 continuity of the curve.

It is interesting to note that the probability distribution for SBX with $n = 0$ is similar to the probability distribution calculated in terms of common interval-schemata of previous section. For $n = 0$, all contracting crossovers have a uniform probability and the probability in the expanding crossovers diminishes with the increase in the spread factor. The only discrepancy between these two distributions is that in the SBX operator with $n = 0$, the probability reduces in a nonlinear manner and that in the ideal interval-schema processing, the probability reduces linearly. Nevertheless, we can argue that the SBX operator with $n = 0$ respects interval-schema processing better than the BLX-0.5 operator. However, other SBX operators with nonzero n do not respect interval schema. This analysis suggests that the other SBX operators may respect some other schema which represents contiguous points but with varying importance. The formalization of this schema processing is a mathematical exercise, which would be of little importance to the current study. Thus, we avoid that exercise in this thesis and focus our attention to study of above described crossover only.

2.4.2 Analysis of effect of n over expansion of population

In the previous section, we stated that value of exponent n basically reflects the propensity of GA search. In the following, we give a more formal discussion on this issue. Since spread factor β represents a normalized parameter for the spread of children points, we wish to establish a relation between n and probability that a certain value of β will be exceeded in a set of N randomly generated β values according to SBX probability distribution. For this purpose, let's assume that population size is $2N$, giving rise to total N random instances of β each generated according to SBX distribution. Further, let $p(\beta_0)$ denote the probability that a single random value of β will not exceed a specific value β_0. This probability $p(\beta_0)$ can be obtained by integrating the equations 2.18 and 2.19 from $\beta = 0$ to β_0,

$$p(\beta_0) = \begin{cases} \frac{1}{2}\beta_0^{n+1} & \text{if } \beta_0 \le 1.0 \\ 1 - \frac{1}{2\beta_0^{n+1}} & \text{otherwise.} \end{cases}$$

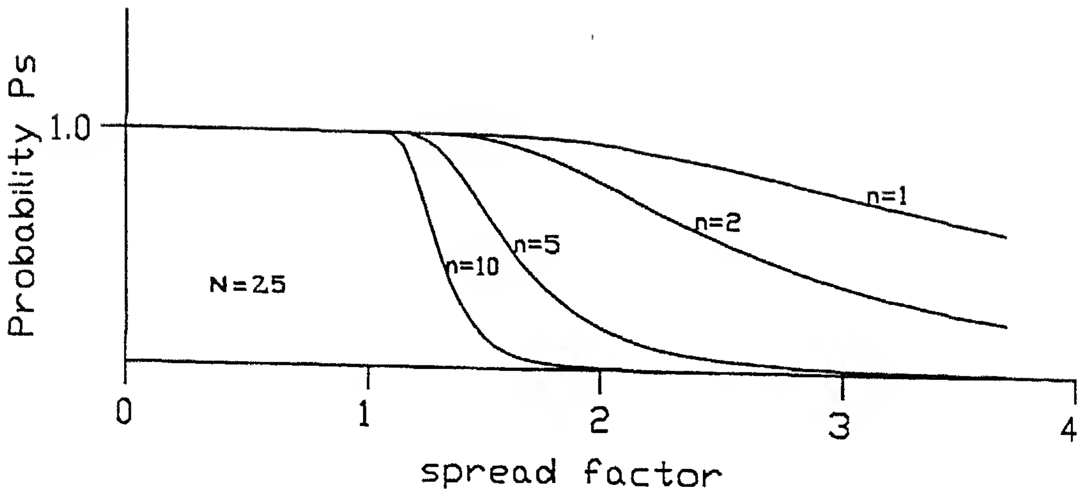

Figure 2.10: Effect of exponent n on probability of spread of β values.

Now, the probability that all N instances of β will be confined to range $(0,\beta_0)$,

$$p_N(\beta_0) = (p(\beta_0))^N$$

since all N events are independent to each other. Further, the probability that at least one β value generated exceeds β_0, will be given by,

$$P_s(\beta_0) = 1 - p_N(\beta_0) = \begin{cases} 1 - \frac{1}{2^N}\beta_0^{N(n+1)} & \text{if } \beta_0 \leq 1.0 \\ 1 - \left(1 - \frac{1}{2\beta_0^{n+1}}\right)^N & \text{otherwise.} \end{cases} \tag{2.20}$$

The equation 2.20 is used to plot the figure 2.10 which shows $P_s(\beta)$ versus β for various values of n. The figure illustrates that for smaller values of n, there are higher chances to generate far-off points (higher β values) whereas low values of n will most probably confine to smaller ranges of β spread.

2.4.3 Problems involving fixed variable bounds

Figure 2.8 shows that the search of SBX crossover extends practically to any point in the real space. This allows the children points to lie anywhere between negative ∞ to positive ∞. In certain problems, the search space may be bounded. For example, if the length of a member in an engineering component design is a variable, negative values of the variable do not make sense. The SBX operator as described above does not always guarantee a child point with a positive value, even though both parents may have positive values. The contracting and expanding probability distributions can be modified so that the probability of creating points outside a

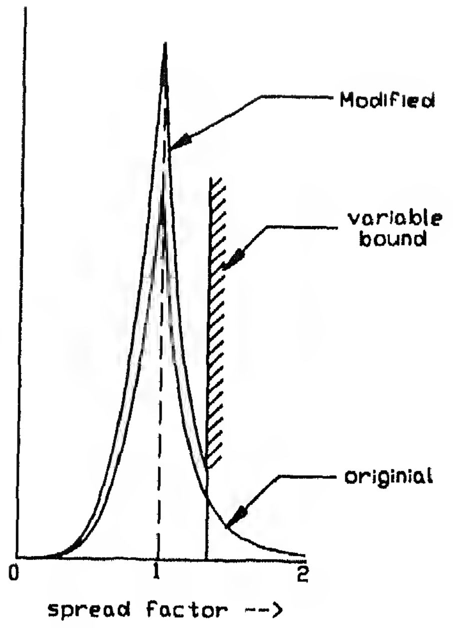

Figure 2.11: Modification of probability distribution curve for rigid bounds on variables.

given bound is zero, while maintaining the same nature of the probability distribution. The way we have achieved it is that for any two given parent points, the cumulative probability of the children points within the given bound is first calculated. If this probability is a (say), the probability distribution function is multiplied by a factor $1/a$, so that the overall probability for points within the bounds is equal to one. Fig. 2.11 shows the distribution curves before and after modification by this strategy.

In the following, we discuss how the SBX operator can be extended to multi-variable optimization problems.

2.4.4 Extension to multi-variable problems

The SBX developed above for single-variable functions can be easily extended for multi-variable problems. There can be various approaches to do this, of which we have implemented the following three approaches :

- *Approach 1 : Simulation of single-point binary crossover :* The most common way to code multiple variables using a binary coding is to use a mapped concatenated string (Goldberg, 1989). In a single-point crossover, one cross site is chosen along the entire string and the contents on the right-side of the cross site are swapped between the parents. When binary-coded GAs are used to solve a N-variable function, and when the cross site falls on the bits representing the k-th variable from the left, the single-point crossover creates a new value of the k-th variable and swap the $(k+1)$-st to N-th variables between the parents. The same operation can be simulated by using the SBX operator on the k-th variable (k is chosen between 1 to N at random) and swapping the $(k+1)$-st to N-th variables between the parents.

- *Approach 2 : Uniform crossover :* It has been shown elsewhere (Booker, 1993) that the feature of the single-point crossover discussed in approach 1 causes a large positional bias for certain combinations of variables to get propagated from the parents to children. Hence in the second approach, we choose a mechanism similar to *uniform* crossover for multiple variables (Syswerda, 1989) for choosing which variables to cross. We perform SBX crossover in each variable with a probability 0.5. This causes about 50% of the variables to be crossed, on an average. Note that a different value of random spread factor will be generated in general, for each variable that is crossed. In the case of single-variable functions, since there is only one variable, the SBX is performed with a probability 1.0.

- *Approach 3: Children on a line :* If we calculate a single random spread factor β and apply it on each variable, the resulting children points will be aligned with the parent points. Even though this strategy creates some bias in the search space, this sort of bias is sometimes useful for certain problems (see table 3.10 in chapter 3).

2.5 Real-coded mutation operator

In real-coded GAs, the mutation operation can be achieved by choosing a neighbouring point in the vicinity of the current point (Back, Hoffmeister, and Schwefel, 1991). Following the same strategy, a real-coded mutation operator is developed on the same lines as SBX operator. This operator is applied on a single parent point (as in traditional GAs) and results in a perturbance of its parameter values. This perturbance is calculated again using a probability distribution which has the property that small perturbations are much more probable than large ones. In order to explain the working of this operator, we first define the *perturbance factor* δ as the ratio of actual perturbance Δ to maximum allowed perturbance Δ_{max} :

$$\delta = \frac{\Delta}{\Delta_{max}} = \frac{x_{new} - x_0}{\Delta_{max}},$$

where x_0 and x_{new} denote the parameter values before and after applying mutation, respectively. Perturbance factor δ is randomly generated using the following probability distribution :

$$\mathcal{P}(\delta) = 0.5(n+1)(1-|\delta|)^n \qquad -1 \leq \delta \leq 1 \tag{2.21}$$

where n is a non-negative number. This distribution is shown in fig 2.12 for $n = 0, 1$ and 4. Fig. 2.12 shows that this distribution is very similar to SBX's contracting crossover distribution for range ($\beta = 0$ to 1) except that both distributions are mirror image to each other. Here also, a low value of n gives chances to far-off children points whereas a high value gives much more probability to near-by points. After a δ value has been calculated according to probability distribution of eq. 2.21, the new value of parameter (x_{new}) can be obtained using eq. 2.5.

$$x_{new} = x_0 + \delta \Delta_{max}$$

Ideally the maximum allowed perturbance Δ_{max} should vary with generations, (namely a large value initially and smaller values for later generations), but in our implementation we have kept a fixed value for Δ_{max} equal to half of the search space for the particular variable being mutated.

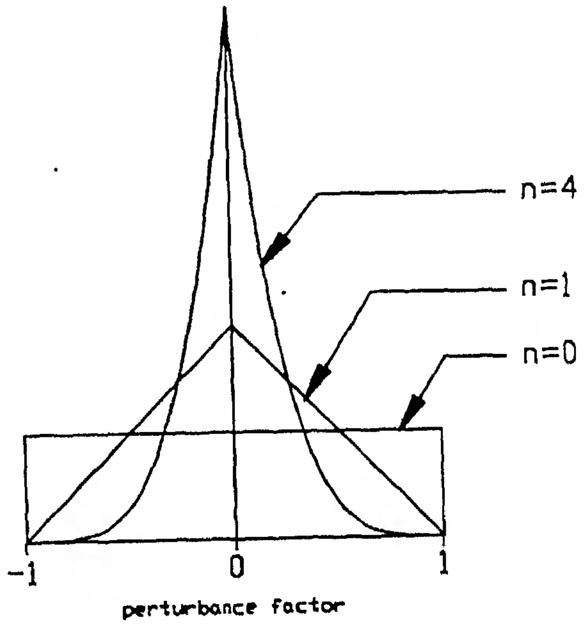

Figure 2.12: The probability distribution of perturbance factor δ.

2.6 Binary GA vs real-coded GA with SBX

In this chapter, we discussed the shortcomings of binary GAs, when applied to continuous search space. A discussion on real-coded GAs was also presented, with a special emphasis on SBX crossover operator developed in this thesis. A real-coded GA using SBX has a number of advantages over binary GAs or other real-coded GAs as outlined below :

1. *No problem of Hamming cliff :* Since real-coded GAs use the parameter values themselves, the question of Hamming cliff does not arise at all.

2. *Adaptive precision level :* Ignoring the fact that computers have limited precision level, we can argue that there is no theoretical limit of the precision level, upto which a real-coded GA can go. Unlike the fixed precision obtained in binary GA, the real-coded GAs give a precision in the solution which is adaptive in the sense that it keeps on refining corresponding to status of convergence of GA.

3. *Flexible variable bounds :* The polynomial distribution used in the SBX crossover lets the children to lie practically anywhere between negative ∞ to positive ∞. This feature of SBX distinguishes it from binary GA or other real-coded crossovers which operate inside fixed variable bounds. In certain situations, where this feature is undesirable, we apply the simple modification discussed in section 2.4.3.

4. *Processing and preservence of useful schemata :* It was discussed in section 2.4 that SBX with $n = 0$ processes interval schemata. SBX with other value of n may respect some other schemata which represent continuous regions. A probability distribution used for SBX preserves more number of interval schemata than BLX, in general. This can be realized by following consideration. Let's assume that two parent points p_1 and p_2 are used to generate children points c_1 and c_2. If child points are created exactly at middle of parent points (which corresponds to β =0), there is maximum destruction of interval schemata. SBX with a non-zero n value gives zero probability for this case, whereas BLX gives equal probability for any β value between 0 to 1.

In the next chapter, we present our simulation results on a test-bed of 12 functions, which gives a validation of applicability of SBX.

Chapter 3

Simulation Results of SBX vs Binary Crossover

This chapter presents a summary of simulation results of GA with binary crossover and SBX crossover. For the purpose of testing the efficacy of SBX vs binary crossover, we have chosen a test bed of 12 functions, of which nine are taken from the existing literature. The test functions range from single-variable functions to multi-variable functions, from unimodal functions to multimodal functions, and from deterministic functions to stochastic functions. Finally, one blocked function introduced by Goldberg (1991) is included to investigate the effect of blocking on the performance of the SBX operator. Table 3.1 gives a summary of all the test functions.

3.1 Simulation details

A real-coded genetic algorithm using SBX is implemented in C programming language. Appendix I gives a summary of capabilities of this program. The initial population of variable vectors is created in an initial range of variables supplied by the user. However, in subsequent generations, the population is not guaranteed to be confined to the given range, because the SBX operator has a nonzero (though small) probability of creating children points far away from the parent points. This is advantageous in problems where the location of the true optimum is not known beforehand. We demonstrate this flexibility of real-coded GAs with the SBX operator in test problems by initializing the population far away from the true optimum.[1] In all our simulations, we allow the SBX to create any point in the real space with probability distributions given in equations 2.18 and 2.19. In order to investigate the effect of the probability distribution of contracting and expanding crossovers in the SBX operator, we present

[1] However, this feature of the SBX operator is redundant for problems where the lower and upper limits of the variables are absolutely known. In those cases, the modification as discussed in section 2.4.3 can be used.

Table 3.1: A summary of the test functions

S.No.	Function Name	Number of variables	Comments
1.	V-function	1	unimodal, smooth, continuous
2.	V-cliff function	1	unimodal, smooth, discontinuous at optima
3.	Bimodal, equal spread	1	bimodal, smooth equal spread of both basins
4.	Bimodal, unequal spread	1	bimodal, smooth global basin narrower
5.	Pole problem	2	four optima, global basin narrow
6.	De Jong's F1	3	smooth, unimodal
7.	De Jong's F2	2	multimodal, smooth
8.	De Jong's F3	5	discrete, stair-case type, unimodal
9.	De Jong's F4	30	unimodal, stochastic (noisy)
10.	De Jong's F5	2	25 optima, global basin narrow
11.	Rastrigin's function	20	10^{20} optima, one is global
12.	Blocked function	2	multimodal, smooth, global optimum blocked by local ones

simulation runs with a number of different exponent values (n). It may be recalled here that for small values of the exponent n, the children points far away from the parent points can be obtained, but for large n, only points closer to the parent points are likely to be generated. To investigate primarily the effect of crossover alone, we have kept mutation probability zero.

It is important to mention here that the performance of GA can not be measured truely by a single run. Since GAs use probabilistic approach in it's search process, it is possible that even with the specification of the same GA parameters, the performance of GA run may change substantially by initialization with a different population. In GA literature, the performance of GA is measured usually by running the algorithm several times, each time with different initial population and reporting it's *reliability* which is the ratio of number of successful runs and the number of total runs (Eshelman and Schaffer, 1993; Gordon and Whitley, 1993). We follow the same strategy. For each GA run, the termination occurs if any one (or more) of the following criteria is satisfied.

1. The best individual in the population is close to the true optimal point with a small termination margin ϵ in all problem variables. In this case, we assume that the GA has found the true optimal point.

2. The function value of the best individual in the population is better than or equal to a target function value f_T. In this case, we say that the GA has obtained the true optimal solution.

3. All individuals in the population have converged to a small neighborhood (within a small margin ϵ in all problem variables) to a point other than the global optimum point. This case is termed as premature[2] convergence.

4. The number of generation has exceeded a specified large number T_{max}. In this case, the GA has not converged to any solution at all.

Unless specified, all simulations use the following GA parameters :

- Maximum generations T_{max} : 200
- Target function value f_T : Function value at global optimum
- Selection scheme : Binary tournament without replacement
- Crossover for multi-variables : Approach 2 of section 2.4.4 (Uniformly 50% variables)
- Crossover probability p_c : 1.0
- Mutation probability p_m : 0.0

The performance of real-coded GAs with the SBX operator is compared with BLX-0.5 and with single-point crossover. Average number of function evaluations required to successfully solve a problem (when either criterion 1 or 2 above is satisfied) is tabulated for each problem. Other GA parameters are mentioned in the description of the simulation results, given in the following.

[2]Note that if GA converges to a small neighborhood of global solution, then criterion 1 or 2 will be satisfied before this.

3.2 V-function

This function is a single-variable function defined in the range $0 \leq x \leq 1$ (Eshelman and Schaffer. 1993) and is shown in figure 3.1. The function has only one minimum point at $x = 0.5$:

$$f_1(x) = |x - 0.5|. \tag{3.1}$$

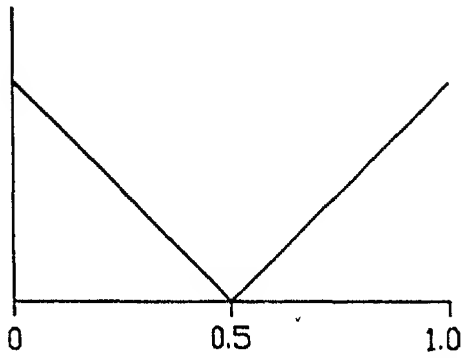

Figure 3.1: Function f_1 (V-function).

First. the binary-coded GAs are used. With a string length of 30 and a population size of 50. binary-coded GAs with the single-point crossover operator find a point with six decimal places of accuracy ($\epsilon = 10^{-6}$) only 23 out of 100 times. In rest of the runs, GAs have converged prematurely to a point close to the true optimum, but not as close as within ϵ from the optimum. The average function value of the best individual in all 100 runs is found to be 0.0015, which is very close to the function value of the true optimum ($f_1^* = 0.0$). It is important to note that the minimum point of the above V-function corresponds to a Hamming cliff in the binary coding used, but with $\epsilon = 10^{-6}$ there are about 2,147 strings representing the region $(0.5 - \epsilon, 0.5 + \epsilon)$. However. the presence of Hamming cliffs is one of the problems of the binary coding, as encountered by other researchers (Davis, 1991b).

Real-coded GAs with the SBX operator are applied next. A summary of the simulation results is tabulated in Table 3.2. A termination margin of $\epsilon = 10^{-6}$ is used. The results show that the real-coded GAs with the SBX operator find the optimum point in all simulations with the chosen tolerance. Notice that the performance of the SBX operator remains the same for different values of the exponent n.

The success rate of BLX-0.5 operator on this problem is equally well to SBX. Since the

Table 3.2: Simulation results of GAs with different crossover operators on the V-function are tabulated. The number in brackets in column six indicates the number of times GAs did not converge to any point at all.

Operator	Range	Popsize	n or ℓ	Performance of 100 runs		Avg. Trials
				Success	Prem. conv.	
1-pt	(0,1)	50	30	23	77	834.8
SBX	(0,1)	50	0	100	0	929.5
			2	100	0	748.5
			5	100	0	818.5
		100	0	100	0	1739.0
			2	100	0	1396.0
			5	100	0	1321.0
BLX-0.5	(0,1)	50		100	0	746.0
		100		100	0	1368.0
SBX	(0.9,1)	50	0	100	0	1279.5
			2	100	0	2018.5
BLX-0.5	(0.9,1)	50		24	76	2604.2
SBX	(0.9999,1)	50	0	100	0	1790.0
			2	100	0	4318.0
BLX-0.5	(0.9999,1)	50		0	61(39)	

SBX operator performs expanding crossovers about the half the time and searches more region than BLX-0.5, the function evaluations required to solve the above problem is less in the case of BLX operator. But, when the same problem is solved with a population initialized in the range (0.9, 1.0), the performance of SBX is much better than that of the BLX-0.5 operator. The success rate of SBX is the same as before, but the success rate of BLX-0.5 reduces to about 25%. Recall that the minimum point does not lie in the range where the initial population is created. In order to find the optimum point, the search operator has to search outside this range. Obviously, the binary-coded GAs will fail to find the true optimum in this case, because of the fixed-mapped coding of the variable. The search power of SBX is further tested by starting with populations confined in the region (0.9999, 1.0000). The termination margin used in this case is also 10^{-6}. With SBX, the expanding crossovers find better points outside the initial range and the whole population migrates towards the true optimum. The success rate of SBX is again 100% in this case, whereas BLX-0.5 performs poorly.

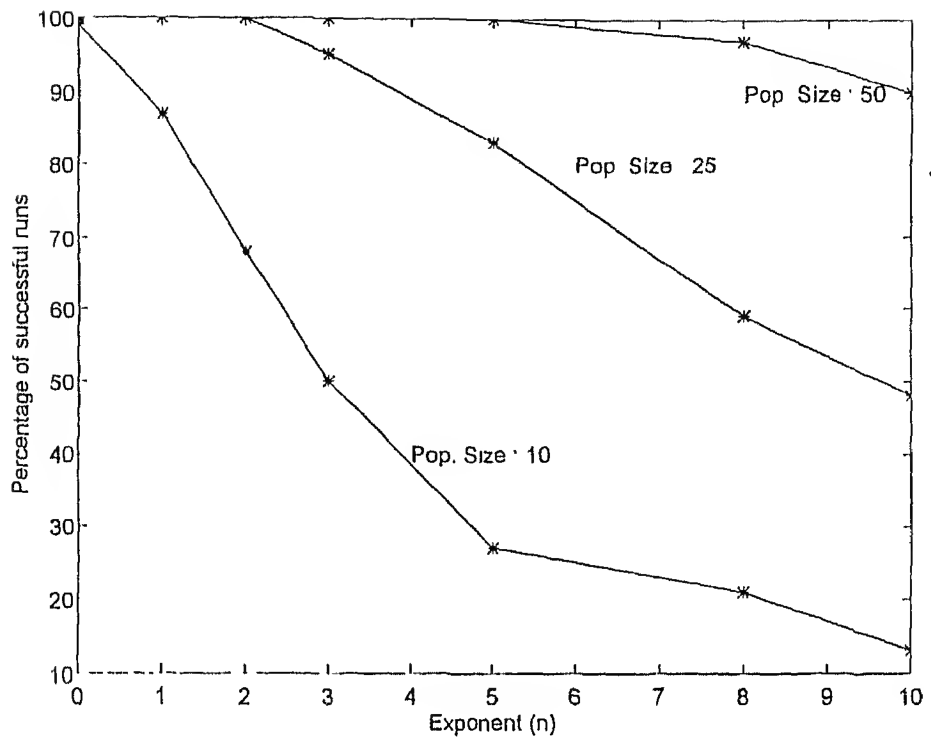

Figure 3.2: Effect of population size and exponent n on performance of GA on V-function.

3.2.1 Parametric study

After gaining the confidence in working of SBX in this simple test function, we began with study of effect of various parameters on the performance of GA. We selected two parameters, population size and tolerance (ϵ) for this purpose. The effect can be seen in the number of successful runs and the average trials given for successful run, which are tabulated in table 3.3 and table 3.4 and plotted in figs. 3.2, 3.3 and 3.4. These figures show that for small values of n, SBX works satisfactorily almost independently to population size and tolerance value. For larger n, the reliability decreases with decrease in population size (fig. 3.2) or with smaller tolerances (fig. 3.3). It is not surprising, because for higher values of n, there are more chances of GA to converge prematurely, and there is likelyhood that the solution is not achieved with desired accuracy. Fig. 3.4 is a plot of tolerance vs average number of trials given for successful runs, for $n = 1$ and population size = 25. The linear nature of curve depicts that if GA can find a solution with some prescribed precision in some particular number of trials, the higher precision in the solution can be obtained with some more function evaluations. This also gives a validation to our claim that real-coded GAs with SBX can goto any arbitrary precision in finding the solution. It is interesting to note that the simulation curve of fig. 3.4 can also be

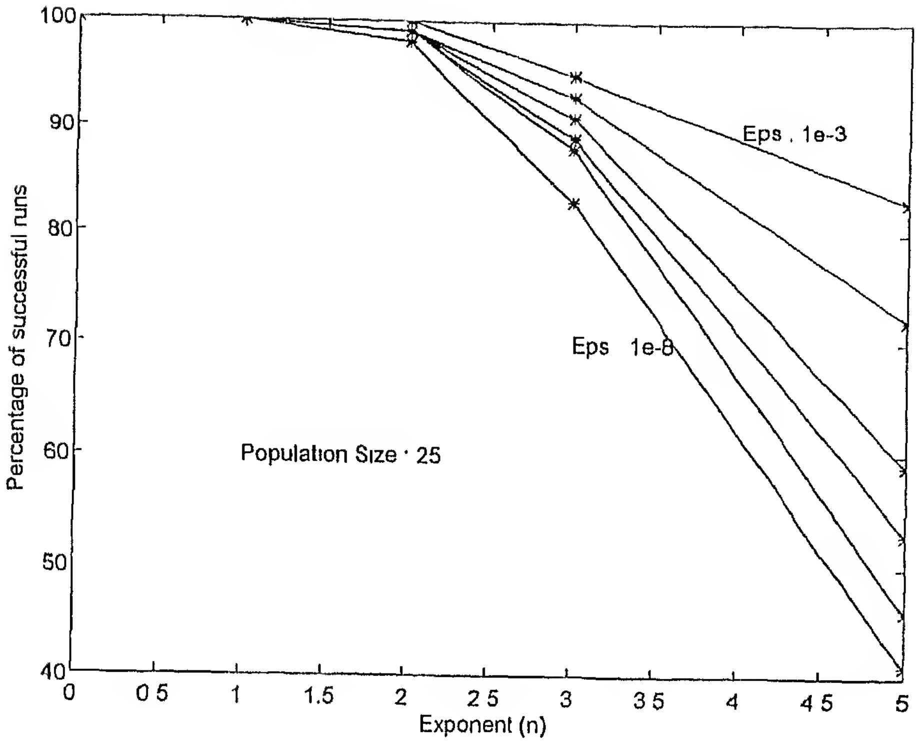

Figure 3.3: Effect of tolerance and exponent n on performance of GA on V-function.

approximated by the following equation.

$$\text{Avg. number of trials } (N_{avg}) = a - b \ \ln \ \epsilon$$

where a and b are some constants. This shows that the computational overheads of the algorithm (here expressed as average number of function evaluations) increase as logarithm of desired accuracy.

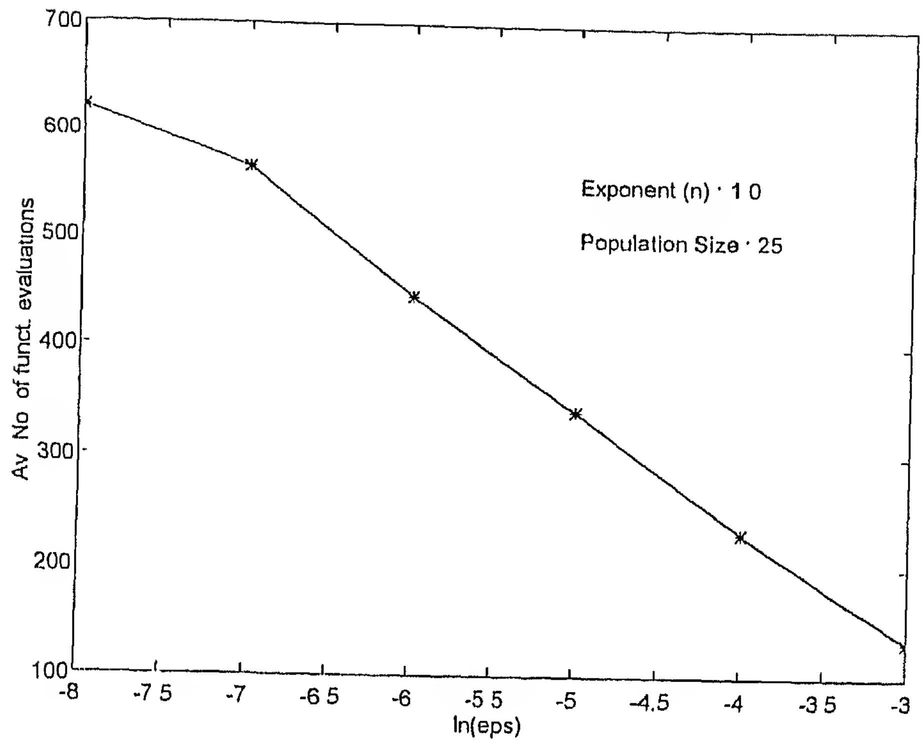

Figure 3.4: Effect of tolerance on average number of evaluations required by GA for V-function.

3.3 V-cliff function

This function is similar to the V function, except that the function has a discontinuity at the minimum point ($x^* = 0.5$) (Eshelman and Schaffer, 1991) and is shown in fig. 3.5:

$$f_2(x) = \begin{cases} 0.6 - x, & \text{if } x < 0.5; \\ x - 0.5, & \text{otherwise.} \end{cases} \tag{3.2}$$

The tolerance parameter $\epsilon = 10^{-6}$ is used again. Table 3.5 shows the performance of binary-coded GAs with the single-point crossover, real-coded GAs with SBX and BLX-0.5 on this function.

Table 3.3: Simulation results of GAs with different population sizes on the V-function.

Tolerance (ϵ)	Range	Popsize	n	Performance of 100 runs		Avg. Trials
				Success	Prem. conv.	
10^{-3}	(0,1)	10	0	99	1	93.7
			1	87	13	90.5
			2	68	32	73.8
			3	50	50	73.0
			5	27	73	54.8
			8	21	79	40.5
			10	13	87	39.2
		25	0	100	0	143.0
			1	100	0	132.0
			2	100	0	131.0
			3	95	5	130.8
			5	83	17	146.4
			8	59	41	133.9
			10	48	52	109.4
		50	0	100	0	222.0
			1	100	0	206.0
			2	100	0	183.5
			3	100	0	217.5
			5	100	0	224.5
			8	97	3	211.3
			10	90	10	244.9

The binary-coded GAs with single-point crossover is not able to find a solution close to the true optimum in 100% simulations. The performance of binary-coded GAs is worse than the simple V-function. The real-coded GAs with the SBX operator, on the other hand, performs well for small values of exponent n. The deterioration in the performance of SBX with either $n = 0$ and large n is due to the discontinuity at the optimum point. However, the performance of BLX-0.5 is 100% with slightly more function evaluations than that of SBX with $n = 2$. The table clearly shows that SBX performs much better than BLX-0.5 when the population is initialized far away from the minimum point.

3.4 Bimodal, equal spread function

The bimodal function used here has two minimum points—one is better than the other (fig. 3.6). The basin of attraction of each minimum point is identical to the other:

$$f_3(x) = \begin{cases} -\exp(-(x-0.25)^2/0.01), & \text{if } x \le 0.5; \\ -0.5\exp(-(x-0.75)^2/0.01), & \text{otherwise.} \end{cases} \tag{3.3}$$

Table 3.4: Simulation results of GAs with different tolerances on the V-function.

Tolerance (ϵ)	Range	Popsize	n	Performance of 100 runs		Avg. Trials
				Success	Prem. conv.	
10^{-4}	(0,1)	25	0	100	0	263.5
			1	100	0	231.5
			2	99	1	220.7
			3	93	7	254.0
			5	72	28	231.9
10^{-5}	(0,1)	25	0	100	0	396.5
			1	100	0	340.5
			2	99	1	325.8
			3	91	9	365.7
			5	59	41	326.7
10^{-6}	(0,1)	25	0	100	0	536.3
			1	100	0	444.0
			2	99	1	434.6
			3	89	11	461.2
			5	53	47	438.7
10^{-7}	(0,1)	25	0	100	0	667.8
			1	100	0	563.0
			2	99	1	543.7
			3	88	12	577.6
			5	46	54	535.9
10^{-8}	(0,1)	25	0	100	0	747.5
			1	100	0	622.0
			2	98	2	618.4
			3	83	17	647.9
			5	41	59	595.1

This function has a local minimum at $x = 0.75$ and a global minimum at $x = 0.25$. The function values at the local and global minima are -0.5 and -1.0 respectively. This function is chosen to test whether real-coded GAs with SBX can find the global minimum point or not. Table 3.6 compares the performance of both GAs for a termination margin of $\epsilon = 10^{-4}$. Once again, real-coded GAs with both SBX and BLX-0.5 perform 100% of the time. Despite the presence of the local optimum, real-coded GAs find the true optimum point with the desired accuracy. In binary-coded GAs, a larger population is required to achieve a desired solution accuracy for a problem coded in a bigger string (Goldberg, Deb, and Clark, 1992). However, the performance of binary-coded GAs is inferior to that of real-coded GAs with SBX or BLX-0.5.

The performance of neither real-coded GAs is good for runs with adverse initial populations

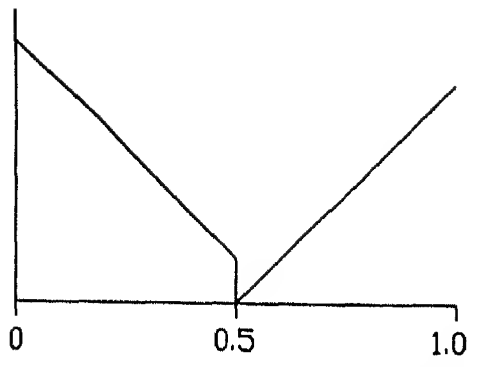

Figure 3.5: Function f_2 (V-cliff function).

Table 3.5: Simulation results of GAs with different crossover operators on the V-cliff function are tabulated. The number in brackets in column six indicates the number of times GAs did not converge to any point at all.

				Performance of 100 runs		
Operator	Range	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
1-pt	(0,1)	50	30	16	84	840.6
SBX	(0,1)	50	0	88	0(12)	2,015.9
			2	100	0	1,382.0
			5	58	42	1,191.4
		100	5	96	4	2279.2
BLX-0.5	(0,1)	50		100	0	1,726.0
SBX	(0.9999,1)	50	0	88	0(12)	2,795.5
			1	100	0	3,253.0
			2	74	2(24)	7,147.3
BLX-0.5	(0.9999,1)	50		0	62(38)	

(with a population initialized in the local basin). Since, the population is initialized in the range $(0.5, 1.0)$, the global minimum point ($x^* = 0.25$) is not included in the range. Thus, the binary-coded GAs will not be able to find this minimum point with the above initial populations. The performance of the real-coded GAs with SBX is marginally better than with BLX-0.5. Since about 50% of the points are created outside the range enclosed by the parents, a few points in the global basin are expected to be created by SBX. Since the binary tournament selection scheme and a $p_c = 1$ are used in the above simulations, the selection pressure is not enough to maintain those points in the global basin. When the crossover probability of SBX is reduced to $p_c = 0.5$, the disruption of good points reduces and the success rate of SBX increases to 34 out of 50 runs with $n = 0$ (with approximately 2,006 function evaluations).

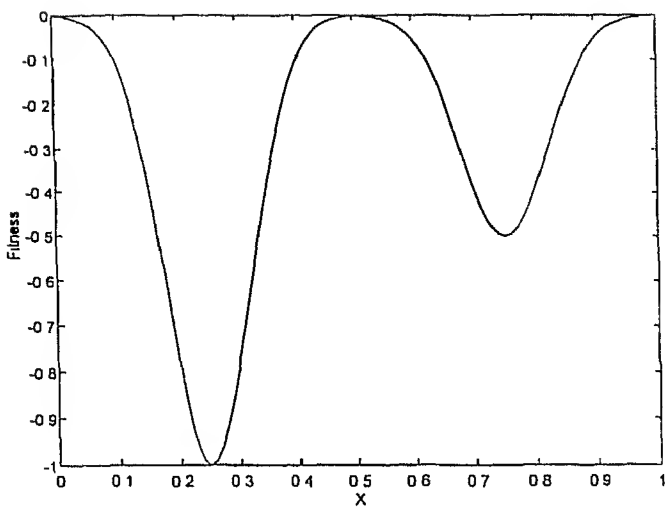

Figure 3.6: Function f_3 (bimodal, equal spread function).

3.5 Bimodal, unequal spread function

This function is a modification of the previous function (fig. 3.7). Here, the basin of attraction of the global minimum is smaller than that of the local minimum:

$$f_4(x) = \sum_{i=1}^{2} - \exp(-(x - a_i)^2/b_i), \tag{3.4}$$
$$a_i = (0.2, 0.6) \qquad b_i = (0.0016, 0.16)$$

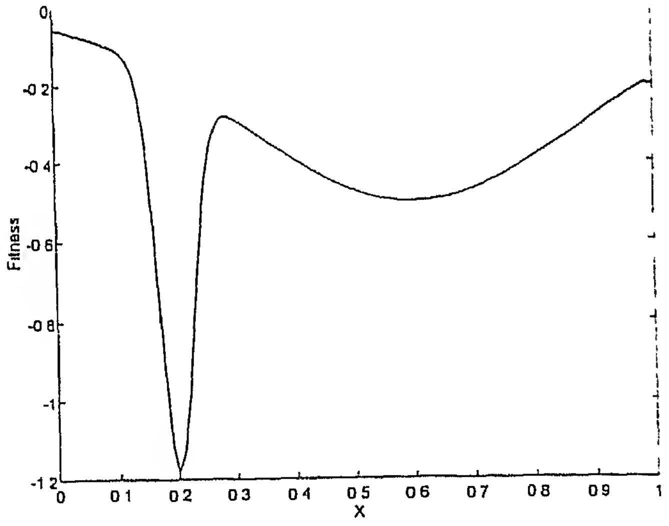

Figure 3.7: Function f_4 (bimodal, unequal spread function).

Table 3.7 compares the performance of single-point crossover, SBX, and BLX-0.5 on this function. Compared to the previous function, the performance of the binary-coded GAs with single-point crossover in this function is marginally worse because the global basin is compara-

Table 3.6: Simulation results of GAs with different crossover operators on the function f_3.

Operator	Range	Popsize	n or ℓ	Performance of 50 runs		Avg. Trials
				Success	Prem. conv.	
1-pt	(0.0, 1.0)	100	20	49	1	765.3
			25	48	2	875.0
			50	30	20	1023.3
SBX	(0.0,1.0)	100	0	50	0	870.0
			1	50	0	688.0
			2	50	0	660.0
			3	50	0	624.0
			5	50	0	642.0
			10	50	0	680.0
			15	47	3	725.5
			20	43	7	700.0
BLX-0.5	(0.0,1.0)	100		50	0	670.0
SBX	(0.5,1.0)	100	0	4	46	4500.0
			2	9	41	1400.0
			5	1	49	1800.0
BLX-0.5	(0.5,1.0)	100		0	50	

tively smaller. The performance of the real-coded GAs with SBX operator remains unchanged except that the SBX with a small exponent n does not work that well. The reason for its failure is the same as that in the previous function with adverse initial population. Although some points in the global basin are created, the selection pressure is not sufficient to maintain them in subsequent generations.

The difference in performance of SBX and BLX-0.5 is clear in this function. Since the local basin occupies most of the search space, a large portion of the initial random population is created in the local basin. About 45 out of 50 GA simulations with BLX-0.5 get confined to the local basin and finally converge to the local optimal solution. Even though some points are created in the global basin, they are difficult to maintain in the population, because of the large disruption rate associated with the BLX-0.5 operator. On the other hand, with SBX having a moderate n, the spread of the children points created using SBX is not substantial. The population gradually shifts towards the global basin and finally converges to the global optimum point.

Table 3.7: Simulation results of GAs with different crossover operators on the function f_4 are tabulated. The number inside brackets in column six indicates the number of times the GA did not converge to any point at all.

Operator	Range	Popsize	n or ℓ	Performance of 50 runs		Avg. Trials
				Success	Prem. conv.	
1-pt	(0,1)	100	20	45	5	817.8
			25	32	18	762.5
SBX	(0,1)	100	0	8	0(42)	450.0
			2	40	6(4)	1117.5
			5	50	0	730.0
			10	50	0	688.0
			20	47	3	663.8
BLX-0.5	(0,1)	100		4	45(1)	1575.0

3.6 Pole problem

The pole problem is a two variable function having four maximum points, of which one is the global maximum. The function is hypothetically constituted to have a light pole at each maximum point with certain illumination capability. The illumination is maximum at the pole and diminishes away from the pole. When multiple poles are present in the search space, the mixed illumination results. For a given distribution of illumination levels in the search space, the objective of the problem is to find the point with maximum illumination. The problem can be written mathematically as follows (fig. 3.8):

$$\text{Maximize} \quad f_5(x,y) = \sum_{i=1}^{4} \frac{c_i h_i}{[h_i^2 + (x-x_i)^2 + (y-y_i)^2]^{3/2}}, \tag{3.5}$$

where (x_i, y_i) represents the coordinates of the poles, c_i represents the intensities of light at the i-th pole, and h_i represents the height of the i-th pole. In the parlance of optimization, the parameter, (x_i, y_i) represents the i-th optimum point, c_i/h_i^2 signifies the function value at the i-th peak, and h_i signifies basin of attraction of the i-th peak. In the above problem, there are four peaks with the following parameter values:

	i			
	1	2	3	4
x_i	0.4	0.3	0.7	0.8
y_i	0.3	0.7	0.2	0.8
c_i	1.0	1.0	1.0	1.125
h_i	0.1	0.1	0.1	0.075.

The optimization problem has a global maximum point at $x^* = (0.8, 0.8)$. Table 3.8 shows

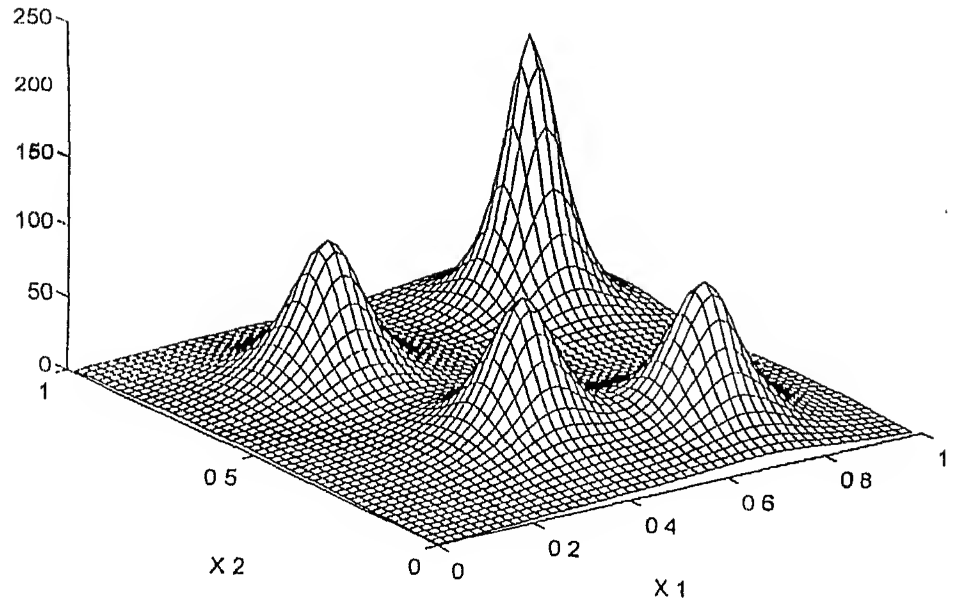

Figure 3.8: Function f_5 (pole problem).

the performance of single-point, SBX, and BLX on this problem for a termination margin $\epsilon = 10^{-3}$.

We observe that real-coded GAs with the SBX operator (for n between 2 to 5) perform as good as the binary-coded GAs with the single-point crossover. However, real-coded GAs with BLX-0.5 do not perform as well as other two GAs.

From the above simulations, we observe that the BLX-0.5 operator performs well only on well-behaved unimodal functions having a continuous search space, but does not perform as well in problems with multiple optimum points. Moreover, the focus in this thesis is to compare the SBX operator used in real-coded GAs with the single-point crossover operator used in binary-coded GAs. Thus, in subsequent simulations, we compare single-point and SBX operators only.

3.7 De Jong's functions

All five De Jong's test functions (De Jong, 1975) are tried next. The simulation results for the single-point and the SBX operators are tabulated in Table 3.9 through table 3.13.

Function I (fig. 3.9) is a three-variable unimodal function having a minimum point at

Table 3.8: Simulation results of GAs with different crossover operators on the function f_5 are tabulated. The number inside brackets in column five indicates the number of times the GA did not converge to any point at all.

			Performance of 10 runs		
Operator	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
1-pt	200	20	6	4	3666.7
		30	9	1	3977.8
		40	8	2	2975.0
SBX	200	0	0	9(1)	
		1	4	6	3,700.0
		2	8	2	3,375.0
		3	9	1	2,866.7
		4	8	2	3,300.0
		5	9	1	3,200.0
BLX-0.5	200		3	7	2,933.3

$(0,0,0)^T$:

$$f_6(x_i) = \sum_{i=1}^{3} x_i^2 \qquad -5.12 \leq x_i \leq 5.12 \tag{3.6}$$

In real-coded GAs with the SBX operator, points outside the specified range may also be created. Thus, we assume that the function is also defined outside the above range, but the initial population is created in the above range. The simulation results (tabulated in Table 3.9) show that GAs with SBX are able to find the minimum point with two decimal places of accuracy in all simulations for small values of n. The binary-coded GAs with the single-point crossover find the minimum point in only 50% of the simulations. The results for SBX is expected because the function is unimodal and continuous in the real search space. In binary-coded GAs, however, the minimum point falls in a Hamming cliff, which causes difficulty to solve the problem in 100% simulations.

Function II (fig. 3.10) is a two-dimensional minimization problem with the global minimum at $(1,1)^T$:

$$f_7(x_1,x_2) = 100(x_1^2 - x_2)^2 + (1-x_1)^2 \qquad -2.048 \leq x_1, x_2 \leq 2.048 \tag{3.7}$$

For real-coded GAs, the initial population is created in the specified range. The results in Table 3.10 shows that both GAs can not find the optimum point $(1,1)^T$. In this function, the slope towards the optimum point is very small and the population gets converged to a point along the ridge. When the SBX operator is performed along a line joining the two parents

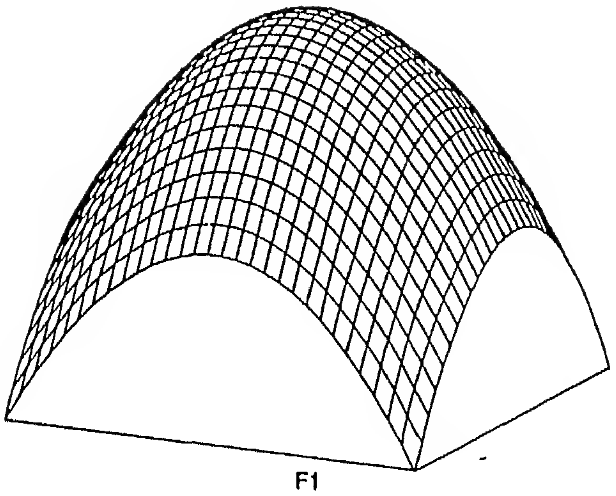

Figure 3.9: Inverted, two-dimensional version of De Jong's (1975) test function F1.

(approach 3 of section 2.4.4 of previous chapter), near 100% convergence is observed for $n = 1$ to 2.

Function-III (fig. 3.11) is a five-dimensional stair-case like function where the function value is monotonically nondecreasing as the variables are increased :

$$f_8(x_i) = \sum_{i=1}^{5} \text{integer}\,(x_i) \quad -5.12 \le x_i \le 5.12 \tag{3.8}$$

Since the SBX operator may create children points outside the specified range, we have modified the original function by taking the absolute value of the function creating the initial population in the range $0 \le x_i \le 10.24$. This results in a minimum point at $x_i = 0$. In order to make the comparison fair, binary GAs are also run on this modified function in the range $0 \le x_i \le 10.24$. Both GAs perform equally well on this function. It is interesting to note that despite the discontinuity of the function at many points in the search space, the real-coded GAs with SBX are able to find the true optimum. This is because the function is monotonically nondecreasing towards the optimum point and because the population-approach of GAs does not allow them to get stuck at any of the plateau.

Function-IV (fig. 3.12) is a 30-dimensional unimodal function with a zero-mean Gaussian noise :

$$f_9(x_i) = \sum_{i=1}^{30} i x_i^4 + \text{Gauss }(0,1) \quad -1.28 \le x_i \le 1.28 \tag{3.9}$$

The optimum point of the deterministic function is $x_i = 0$ ($i = 1$ to 30) with a function value equal to zero. But due to the addition of the Gaussian noise, the optimum point and hence the

Table 3.9: Simulation results of GAs with different crossover operators on De Jong's function-I.

					Perform. of 50 runs		
Range	Tolerance	Operator	Popsize	n or ℓ	Success	Premature conv.	Avg. Trials
(-5.12,5.12)	10^{-2}	1-pt	100	30	25	25	2532.0
				45	26	24	2500.0
				60	19	31	3254.8
		SBX	100	0	50	0	2556.0
				1	50	0	2124.0
				2	50	0	2100.0
				3	50	0	2124.0
				5	48	0	2183.3
				10	24	26	2420.8

Table 3.10: Simulation results of GAs with different crossover operators on De Jong's function-II are tabulated. The number inside brackets in column seven indicates the number of times the GA did not converge to any point at all.

					Perform. of 50 runs		
Approach of multi-var.	Tolerance	Operator	Popsize	n or ℓ	Success	Premature conv.	Avg. Trials
	10^{-3}	1-pt	100	24	0	50	
				30	0	50	
				40	0	50	
Approach-2		SBX	100	0-20	0	50	
Approach-3		SBX	100	0	40	0(10)	4400.0
				1	50	0	3334.0
				2	48	2	2920.8
				3	41	9	2770.7

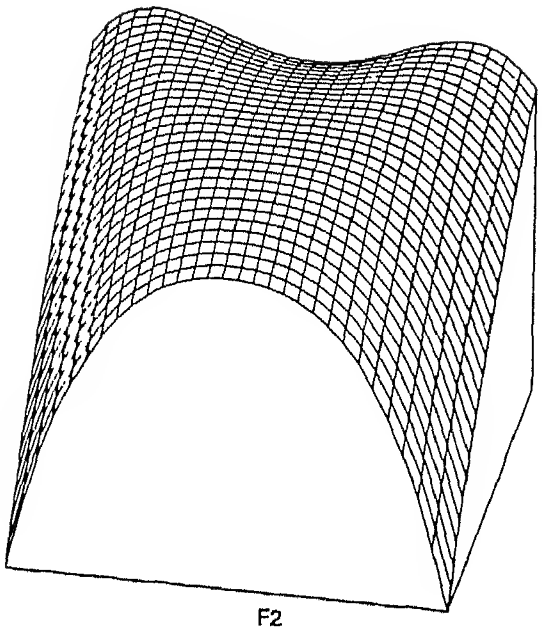

Figure 3.10 : Inverted, De Jong's (1975) test function F2.

optimum function value changes. To decide the convergence to the optimum, we have assumed a target function value of $f_T = -3.0$. If any point has a function value smaller than f_T, the simulation is terminated. A termination margin of $\epsilon = 0.16$ is used. Real-coded GAs with the SBX operator perform much better than the binary-coded GAs for $n = 2$ to 5.

Function-V (fig. 3.13) is a 2-dimensional, multimodal function having 25 minimum points, of which only one is the global minimum. The basin of attraction of the global minimum point is very narrow, thereby making it difficult for the classical optimization methods to solve the

Table 3.11: Simulation results of GAs with different crossover operators on De Jong's function-III.

					Perform. of 50 runs		
						Premature	
Range	Tolerance	Operator	Popsize	n or ℓ	Success	conv.	Avg. Trials
(-5.12,5.12)	1.0	1-pt	100	20	50	0	560.0
				40	50	0	696.0
				50	50	0	720.0
		SBX	100	0	50	0	1256.0
				1	50	0	892.0
				3	50	0	748.0
				5	50	0	722.0

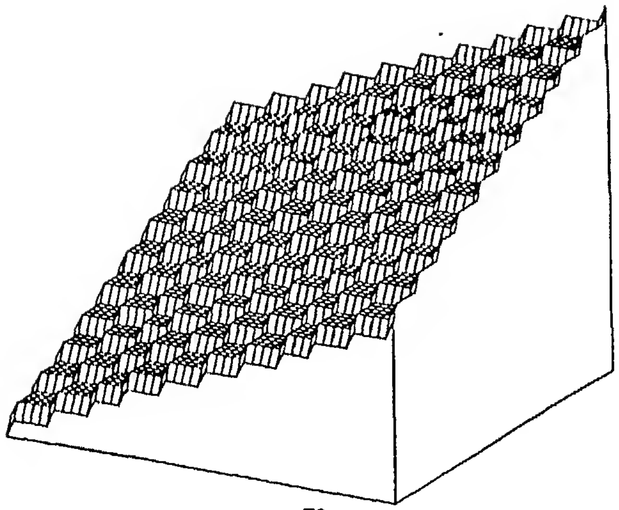

Figure 3.11: Inverted, two-dimensional version of De Jong's (1975) test function F3.

problem to global optimality :

$$f_{10}(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6} \qquad -65.536 \le x_i \le 65.536 \qquad (3.10)$$

Binary-coded GAs with the single-point crossover perform better on this function than real-coded GAs with the SBX operator. This is so because the location of the optimum points are such that the binary coding favors the search process. When the locations of 25 optima are randomly placed, binary-coded GAs succeed in only 64% of the simulations, whereas real-coded GAs with SBX succeed in 54% of the simulations.

From these simulations we conclude that real-coded GAs with SBX has performed either better than or similar to binary-coded GAs in De Jong's test functions. In order to bolster our conclusions further, we apply SBX operator to two more functions.

3.8 Rastrigin's function

Rastrigin's function (fig. 3.14) was used by a number of GA researchers in the recent past (Muhlenbein, Schomisch, and Born, 1991; Gordon and Whitley, 1993). The function has 20 variables and contains 10^{20} local minimum points, of which only one is the global minimum point:

$$f_{11}(x) = 200 + \sum_{i=1}^{20} x_i^2 - 10\cos(2\pi x_i), \quad -5.12 \le x_i \le 5.12. \qquad (3.11)$$

Table 3.12: Simulation results of GAs with different crossover operators on De Jong's function-IV are tabulated. The number inside brackets in column six indicates the number of times the GA did not converge to any point at all.

				Perform. of 50 runs		
Tolerance	Operator	Popsize	n or ℓ	Success	Premature conv.	Avg. Trials
0.16	1-pt	100	240	4	42(4)	15200.0
	SBX	100	0	0	0(50)	
			1	0	0(50)	
			2	50	0	17032.0
			3	50	0	12454.0
			4	50	0	10274.0
			5	50	0	9372.0

Table 3.13: Simulation results of GAs with different crossover operators on De Jong's function-V are tabulated. The number inside brackets in column seven indicates the number of times the GA did not converge to any point at all.

					Perform. of 50 runs		
T_{max}	Tolerance	Operator	Popsize	n or ℓ	Success	Premature conv.	Avg. Trials
500	0.1	1-pt	100	20	48	2	756.2
				30	50	0	744.0
				40	49	1	836.7
		SBX	100	0	7	2(41)	5014.3
				1	26	8(16)	1492.3
				3	31	17(2)	1154.8
				5	36	14	1097.2
				8	39	11	892.3
				10	37	13	891.9

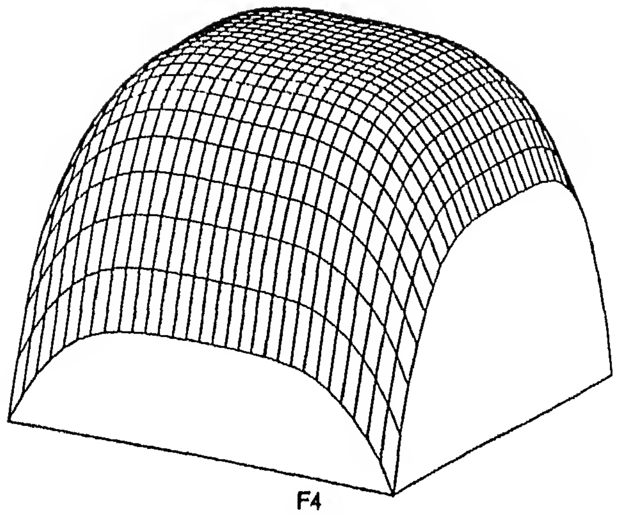

Figure 3.12: Inverted, two-dimensional version of De Jong's (1975) test function F4.

The global minimum point is at $x_i = 0$ for all $i = 1, 2, \ldots, 20$. At this point, the function value is zero. The other local minima occur when the cosine term is one. This function is a difficult optimization problem, because there are 2^{20} different local optimal points (surrounding the global optimum) with function values very close to that at the global optimal point. Since it is relatively a more complex problem, we allow maximum number of generations T_{max} equal to 500. We set $\epsilon = 0.1$ and the target function value $f_T = 1.0$. This allows a termination of the simulation when at least 19 out of 20 variables have converged to their global minimum values.

Binary-coded GAs with equal substring length for all variables are applied first. In all ten simulations, binary-coded GAs prematurely converge to a wrong solution (Table 3.14). Real-coded GAs with the SBX operator perform better than binary-coded GAs. The average function values of the best solution in all simulations are in general better compared to that obtained for binary-coded GAs. The success of real-coded GAs in this problem is due to the controlled allocation of children points under the SBX operator. With moderate n, the children points are not far away from the parent points. So the search slowly progresses towards the optimum point. It is worth mentioning here that with a tolerance of 0.1 in each of 20 variables, a enumerative search method requires about $[(5.12 - (-5.12))/0.1]^{20}$ or $1.48(10^{40})$ function evaluations. Real-coded GAs took only a fraction of this search space to locate a near global minimum point. These results show the efficacy of real-coded GAs with the SBX operator in solving multi-variable, multi-modal functions.

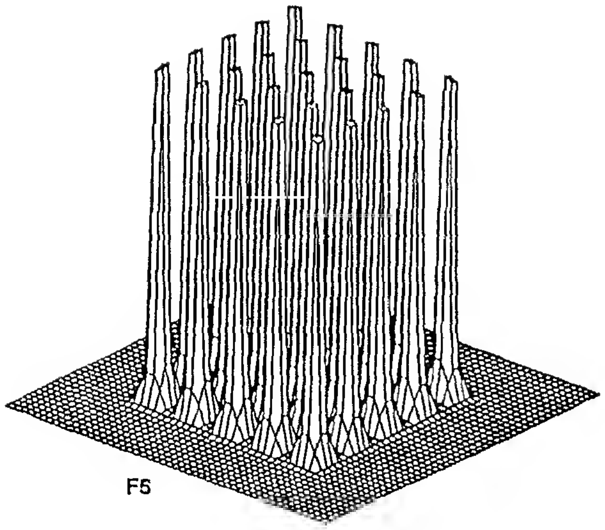

Figure 3.13 : Inverted, De Jong's (1975) test function F5.

3.9 Blocked function

A two-dimensional blocked function (fig. 3.15) is designed according to Goldberg's (1991) suggestion. The mean (over x_1) and global slice (at x_1^*) of the function are plotted in figure 3.16.

The global maximum point lies at the point $(0.4, 0.45)^T$. Two local *hills* are placed on either side of the global optimum. In the global slice (taken at $x_1 = 0.4$), the global maximum point is surrounded by two local maximum points, but the average slice (averaged on x_1 values) is predominant by the local peaks. It is interesting to note that the search along the variable x_1 is unimodal. Thus, the population is expected to converge at $x_1 = 0.4$. Once most of the population converges on or near $x_1 = 0.4$, the search progresses along the global slice. Since the population is converged to the extreme local peaks, it becomes difficult for classical algorithms to find the global point. The function is given below:

$$f_{12}(x_1, x_2) = -(x_1 - 0.4)^2 + \sum_{i=1}^{5} \frac{a_i}{b_i + r_i(x_1 - 0.4)^2 + (x_2 - c_i)^2}. \tag{3.12}$$

Here, the following parameters are used:

	i				
	1	2	3	4	5
a_i	0.002	0.0025	0.014	0.003	0.0028
b_i	0.002	0.0020	0.003	0.001	0.0010
c_i	0.1	0.9	0.45	0.27	0.65
r_i	0	0	10	10	10

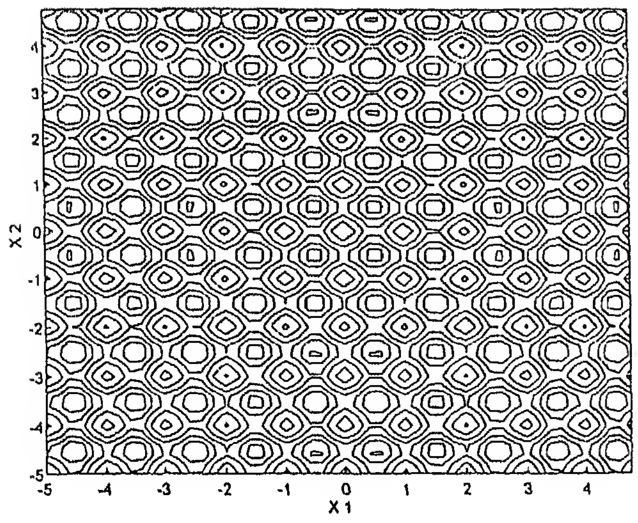

Figure 3.14: Contour plot of 2-dimensional version of function f_{11} (Rastrigin's function). The central basin is the global basin.

Simulation results with binary-coded and real-coded GAs are tabulated in Table 3.15. Although the binary-coded GAs with single-point crossover converge to the global optimum point (with three decimal places of accuracy) only 5 to 15 times, they converge to the global basin in about 70% of the simulations. Better results are observed with the real-coded GAs with SBX; in at least 80% of the runs, they have converged to the global maximum point. In all successful simulations of real-coded GAs, it is observed that the variable x_1 converges to the true optimal value first, as predicted by Goldberg (1991). Subsequently, the search along the global slice continues and finally finds the global maximum point. Whenever real-coded GAs have failed to find the global optimum point, the convergence to the variable x_2 has occurred first. When this happens, the population shifts in any one or both the peaks shown on the mean slice plot in figure 3.16. Since the function on x_1 is unimodal with maximum at $x_1 = 0.4$, the solution converges to a wrong solution. Although the SBX operator works on real parameter values, its search power is similar to that in the single-point crossover. With a moderate n, the population located at two extreme optima improves gradually and can overcome the barrier of the local peaks to finally converge to the global optimum.

Table 3.14: Simulation results of GAs with different crossover operators on the function f_{11} are tabulated. The number inside brackets in column five indicates the number of times the GA did not converge to any point at all.

Operator	Popsize	n or ℓ	Performance of 10 runs		Avg. Trials	Avg. function value
			Success	Prem. conv.		
Binary	400	200	0	10		11.32
		300	0	10		
SBX	400	0	0	0(10)		224.5
		2	10	0	49,680.0	6.81
		3	8	2	51,750.0	2.92
		5	6	4	39,866.7	2.39
		10	10	0	28,000.0	3.77
		15	10	0	27,280.0	3.15
		20	10	0	23,360.0	3.08

Table 3.15: Simulation results of GAs with different crossover operators on the function f_{12} are tabulated. The quantity in bracket indicates in column five the number of times GAs have converged to the global basin, but not to the global optimum.

Operator	Popsize	n or ℓ	Performance of 50 runs		
			Success	Prem. conv.	Avg. Trials
Binary	100	20	5	45(27)	1,480.0
		30	15	35(27)	1,753.3
		40	10	40(22)	1,550.0
		50	6	44(32)	1,616.7
SBX	100	0	42	8(0)	2,323.8
		1	43	7(0)	1,927.9
		2	47	3(0)	1,872.3
		3	43	7(0)	1,795.3
		5	40	10(1)	1,677.5
		10	38	12(6)	3,091.7

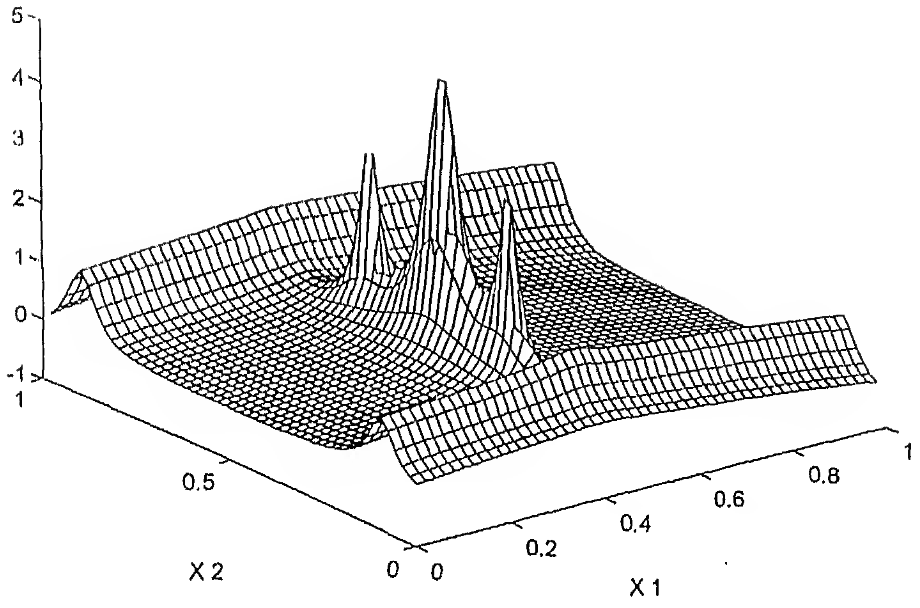

Figure 3.15: Function f_{12} (blocked function).

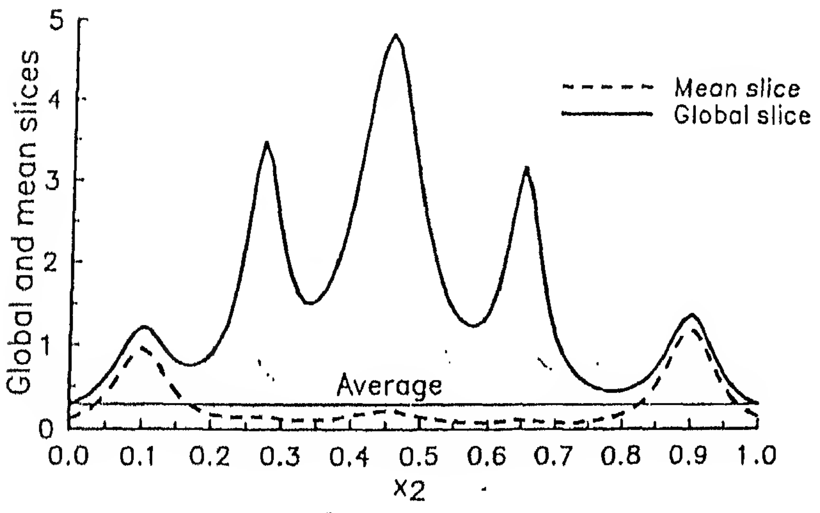

Figure 3.16: The mean slice and global slice (at x^*) of the function f_{12} are shown. The average function value over the search space is 0.287.

3.10 Summary

This chapter presented the simulation results of real-coded GA using SBX vs binary crossover on a number of test functions including De Jong's functions. The results obtained here show that SBX has performed either better or similar to binary crossover in all test functions. It has been found particularly useful in problems, where location of global solution is not known beforehand. Further, SBX has been applied successfully to solve a blocked function introduced by Goldberg (1991). These results present a confidence, with which we begin at applying SBX in an application problem in the next chapter.

Chapter 4

Ambiguous Shape Modelling : Application of GAs

This chapter addresses to the second part of the thesis, i.e. ambiguous shape modelling. Here we discuss the problem and its significance in CAD systems. It is also shown how GA with SBX is applied in this problem.

4.1 The design process and present role of computers in design

Design is a widely used term to denote *the formulation of a plan for the satisfaction of a human need.* According to one definition (Peters et.al., 1990) :

> Design is a creative operation of products, production processes or more generally of humanity serving systems using available materials, products and processes, aiming to achieve a specific function, responding to the demand of the customer and compromising between conflicting constraints as cost, delivery, delay and feasibility.

Any design process normally goes through several phases, which can be broadly classified as (Sandor, 1964):

1. Confrontation with the problem
2. Formulation of the problem
3. Conceptual design of the system
4. Synthesis
5. Generation of analyzable model
6. Experiments, analysis and optimization
7. Presentation

It is to be noted that the above phases are not a linear step-by-step procedure. In a real design process, these steps may be required to be repeated in any order, and the designer needs to go back and forth on these steps.

In a real sense. the present CAD systems are used only for *drafting* and not for *designing* (McCullough, 1990). That means that present computers systems can't assist the designer in the conceptual design and synthesis process. The designer has to first formulate his entire plan and generate an analyzable model (step 5); only then the computer can be used to analyze the design, provide a visual display and optimize it. At this point, the design may already be too constrained so that conceptually different designs can not be represented in the search space and are thereby excluded. This is a serious problem in all design optimization efforts. In the following sections. we focus our attention at the second phase of design (conceptual design) and show how the role of CAD can be extended to even this phase.

4.2 Quantitative vs qualitative models

Almost all the present CAD systems use *quantitative* models to represent object geometries, storing the exact information about its boundary and attributes. In fact, one of the well-established requirement of CAD models (Requicha, 1980) is "unambiguousness", i.e. a single model should represent only a single geometry. However, there are other types of models possible — so-called *qualitative* models, which are less precise in nature. In a typical qualitative model, quantities are represented from a very coarse and functionally relevant discretization such as $(-, 0, +)$, as opposed to the full quantitative specification. This results in the models being *ambiguous*, i.e. a class of objects are represented by a single model.

Qualitative models are important in a number of areas such as function modelling and abstraction, pattern recognition and conceptual design. In this work, we are interested in applying qualitative models in shape modelling. In the next section, we look at some basic techniques on which such a qualitative model can be built.

4.3 Qualitative shape modelling

As stated earlier, qualitative models use qualitative zones to represent parameters of the model. In this thesis, we limit our concern with models of 2-D shapes only. Even though qualitative models are beneficial in a number of areas, their applicability in geometrical applications has remained limited due to problems such as inability to reconstruct the shape for visual display, a problem that can be termed as *inexact visualization* (King and Mukerjee, 1990). Since the model represents not one but many shapes, the user is unable to visualize the shape being described. Hence an appropriate representation scheme should be used to base the qualitative model, such that it is possible to reconstruct the shape. A qualitative model can be built upon different types of representation principles. There are many popular shape modelling techniques (Requicha, 1980) each based on a different principle :

1. Boundary representation in terms of absolute co-ordinates of vertices (Absolute boundary)
2. Boundary representation in terms of lengths of each edge and relative angles made with previous edges (Boundary chain)
3. Storing the pixels (bitmap) that constitute the shape (Uniform grid)
4. Quadtree model (Hierarchical grid)
5. Boolean operations among certain predefined primitives (Set theoretic)
6. Medial Axis Transform (Axis based) (King, 1991; Blum, 1967)

It is to be noted that an appropriate representation should be invariant under geometric transformations, stable and robust. Further, the qualitative version of the model should be able to store the basic abstractions of the shape. At the same time, it should be recognizable, i.e. the system should be able to display at least some of the typical shapes represented by the model. Our objective here is to select a modelling technique in which we should be able to model a class of geometries by representing the various parameters associated with the model, into qualitative zones. All the modelling techniques described above except the last one (axis based) are found to be unstable and hence inappropriate. Medial axis transform has the unique property that it can be redefined to form a valid qualitative model. Recently, work has been done on reconstructable ambiguous shape modelling using qualitative medial axis transform approach (King and Mukerjee, 1990). However these results are limited to convex shapes only and do not present applicability for a general shape. Partly this limitation is due to the difficulty in solving the constraint satisfaction problem using traditional methods, which is why we have chosen to apply the new GA method to this problem. In this work, we concern ourselves only with developing a qualitative medial axis transform (QMAT) with reconstruction and visualization capabilities for handling general shapes including non-convex

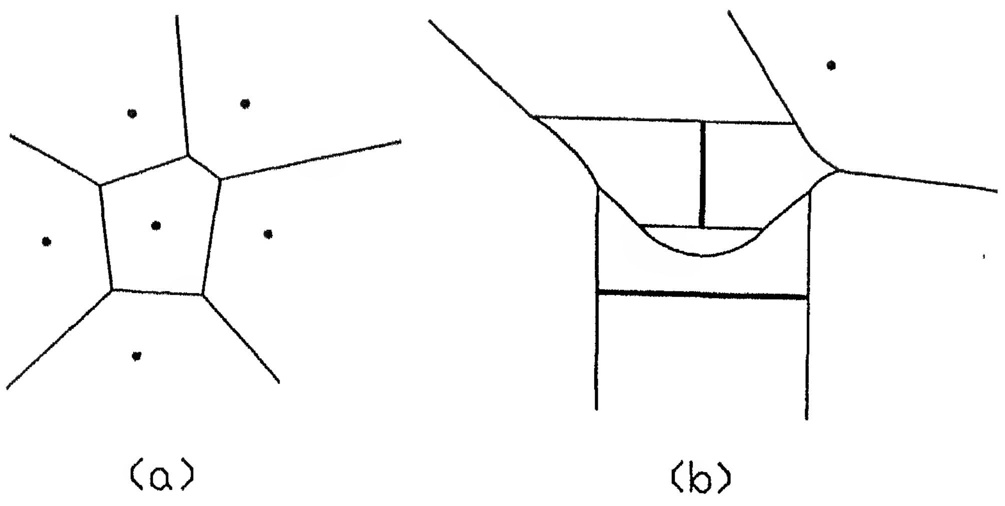

Figure 4.1: Voronoi diagram for (a) point sites, (b) line sites. The thick lines in fig. (b) represent line sites.

ones (Agrawal, Mukerjee and Deb, 1995). The key notion used here relate to that of ***line site voronoi diagram*** which is a topic for the next section.

4.4 Voronoi diagram and medial axis transform (MAT)

Voronoi diagrams have been studied extensively in computational geometry and are used in a range of applications including robotics, geography and facility planning. Given a set of N point sites S_i in a plane, the voronoi diagram of this set is defined as the locus of a point which is equidistant to at least two nearest point sites. The voronoi diagram divides the plane into exactly N regions, each corresponding to one site. These regions have the property that the nearest neighbour of any point in the region R_p will be point site p, where R_p denotes the region corresponding to point site p. For example, Fig. 4.1(a) shows the voronoi diagram for some points. The voronoi diagram of point sites consist of only straight lines. If some of the sites are line segments, the V.D. will, in general, consist of lines and portions of parabolas (Fig. 4.1(b))

Medial axis transform (MAT) of a closed polygon is defined as that part of the voronoi diagram constructed by treating the edges of the polygon as line sites, that lies inside the polygon. It can also be regarded as the locus of the center of inscribed circles that touch at least two edges of a polygon. For all polygons, the MAT is a planar graph consisting of straight lines and parts of parabolas. Various algorithms for computing the MAT are available in the literature (Fortune, 1987; Yap, 1987; Tiwari, 1986). Some sample polygons together with their

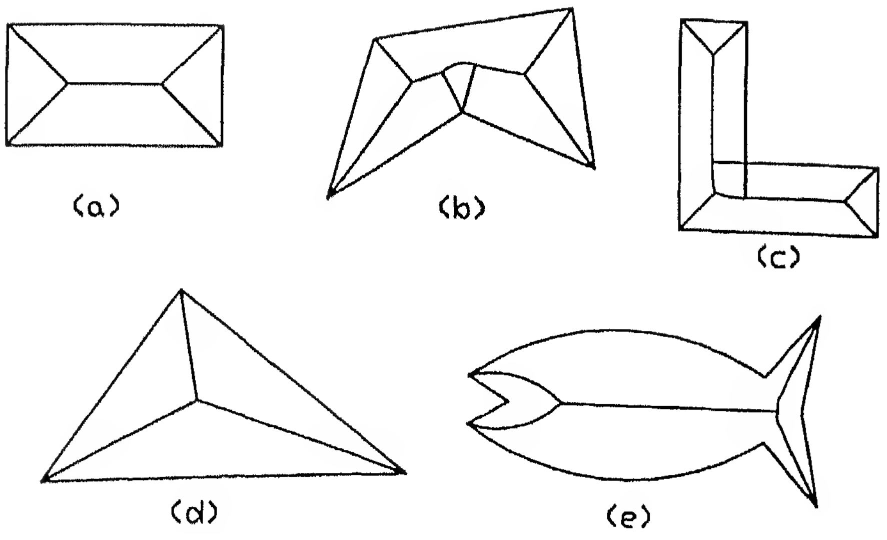

Figure 4.2: Some sample polygons with their MATs.

MAT are shown in fig 4.2.

It is to be noted that mapping from polygon to MAT is one-to-one, i.e. given a MAT, its polygon can be generated uniquely and vice-versa. Hence, the normal MAT can be used to represent the quantitative or exact shape. A detailed description of this application can be found in (Blum, 1967). In this work, we explore the issue of using MATs to represent inexact shapes.

4.5 Hybrid qualitative medial axis transform

An MAT is a planar graph which can be represented by a list of *links* which are segments along the MAT, or fragments of the line-site-voronoi-diagram. Each link is associated with the following information:

- length
- radius function:
- angle with parent

For example, fig. 4.3 shows a typical MAT. AB, BF, BC etc. are different links of MAT which are represented by their end points called *nodes*. Even though the radius function can be of any arbitrary form, we assume linear interpolation function in the current study, thus it will be sufficient to specify the radi only at nodes (start-node-radius, end-node-radius). Further, we define the longest link in the MAT to be *root link*. The $\angle$ of a link is defined by the angle made by this link to it's parent link. For example the angle of link BC is the $\angle CBE$.

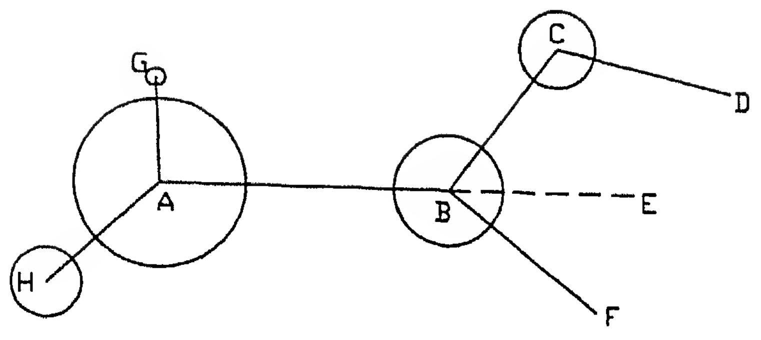

Figure 4.3 : A typical medial axis transform. Note that radii at nodes D and F are zero.

Our objective here is to use the MAT as the basic model on which to apply a qualitative mapping. The qualitative model will then relate all the link lengths and angles using a 2D spatial reasoning model (Mukerjee and Joe, 1990). In the purely qualitative version of the MAT, the lengths of each link in the MAT, their relative angles and their radii are stored as qualitative zones from $(-, 0, +)$ and not as a single value. The problem with such a model is that the discretization, e.g. all negative numbers into a single class, is too coarse, and the resulting geometries are too diffuse to be useful for reconstructing or even usefully constraining the shape. For this reason we use a hybrid qualitative algebra (Mittal and Mukerjee, 1995) in which the qualitative regions are subdivided into regions that are at the right level of discretization for the task. Picking up values from these zones and constructing the polygons accordingly leads to generation of different but qualitatively similar shapes (fig. 4.10). If these uncertainity zones are reduced to a point, the model reduces to a quantitative model. On the other hand, widening these zones results in more diffuse geometries. Note that an appropriate level of discretization depends upon the stage of design; initially a designer will like to have too wide zones of uncertainity to test various possibilities, and as the design tends to be finalised. the level of discretization will be made finer and finer.

In the qualitative version of the MAT, often some links which may not be so important, can get deleted (King, 1991). This usually results in a smoother and simpler shape (fig. 4.4(a)). Also to make the implementation simpler, we assume that all the parabolas of the MAT have been approximated to straight line segments (fig. 4.4(b)).

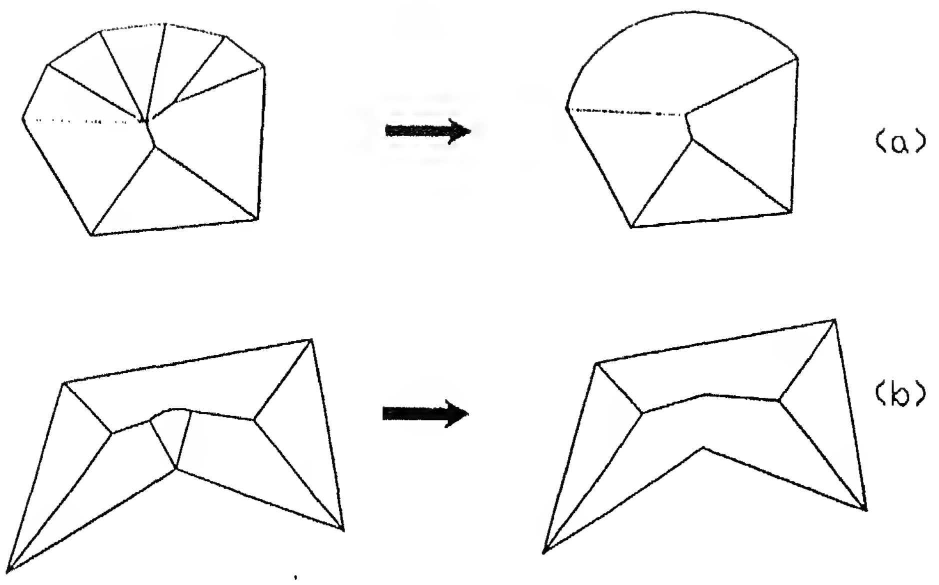

Figure 4.4: Approximations in QMAT (a) Removal of unimportant links, (b) Approximation of parabolas to straight lines.

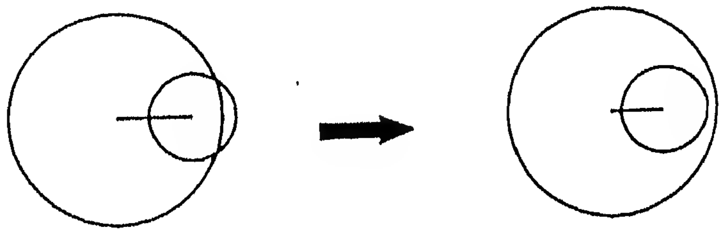

Figure 4.5: A single link MAT becomes infeasible by link length being small.

4.6 Feasibility and consistency of the QMAT

The problem with having inexact zones for the MAT parameters is that these zones may be inconsistent among themselves, i.e. often a randomly generated shape by the model may be inconsistent or infeasible. An MAT is *infeasible* if no boundary can be constructed for these axis specifications. For example, in fig. 4.5 a single link has a very large radius on one side, and it would be impossible to define the boundary. Similarly, fig. 4.6 shows another infeasible case, where a part of boundary is included in the circle at one end. Here also, the boundary is undefined.

An MAT is said to be *inconsistent* if a boundary for the MAT can be generated but the MAT of this boundary is different from input MAT. Consider, for example, the case shown in fig 4.7. Because of the nonconvexity of the shape, there will be some portions of parabolas in the

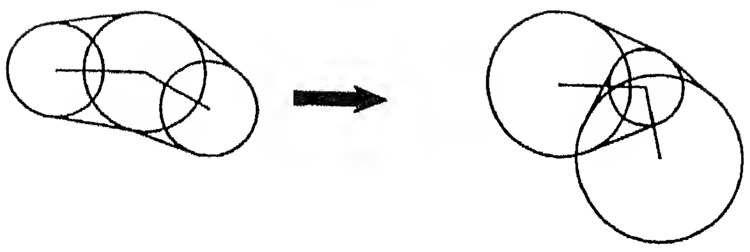
Figure 4.6: Another infeasible MAT because of end radi becoming too high. A part of boundary is included in the circle.

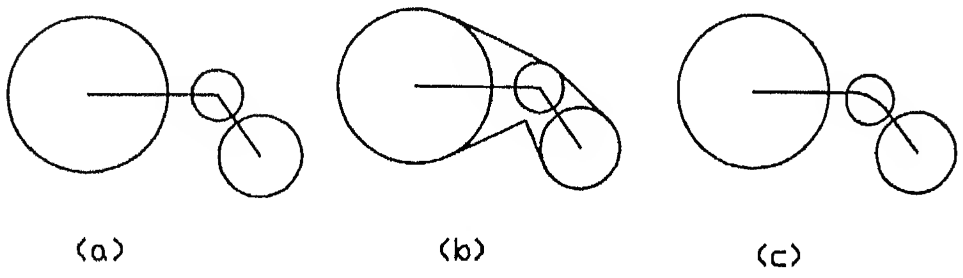

Figure 4.7: An example of inconsistent MAT, (a) Input MAT (b) Reconstruction of contour (c) MAT of generated contour. Note that fig. (a) and (c) are different in the sense that regenerated MAT contains a part of parabola.

MAT, but they have been approximated by straight lines. Thus, a consistent boundary can't be generated by this MAT, because the resulting polyarc would have an MAT that is different from that which was input. However, the parameters of the MAT being varied independently to each other, there is very little control on the contour and such inconsistencies are hard to detect.

To eliminate such inconsistent / infeasible shapes and also to generate desirable shapes, we have used penalty functions coupled with real-coded GA using SBX. Each individual of GA represents an instance of shape generated by QMAT model. Thereafter, it is evaluated in terms of its fitness criteria.

4.7 Reconstruction from the MAT

Once a feasible MAT has been found, its corresponding boundary representation can be obtained by taking the envelope of the circles of appropriate radii. In our case, since these radii are linearly interpolated between the endpoints, this envelope is the tangent between the two

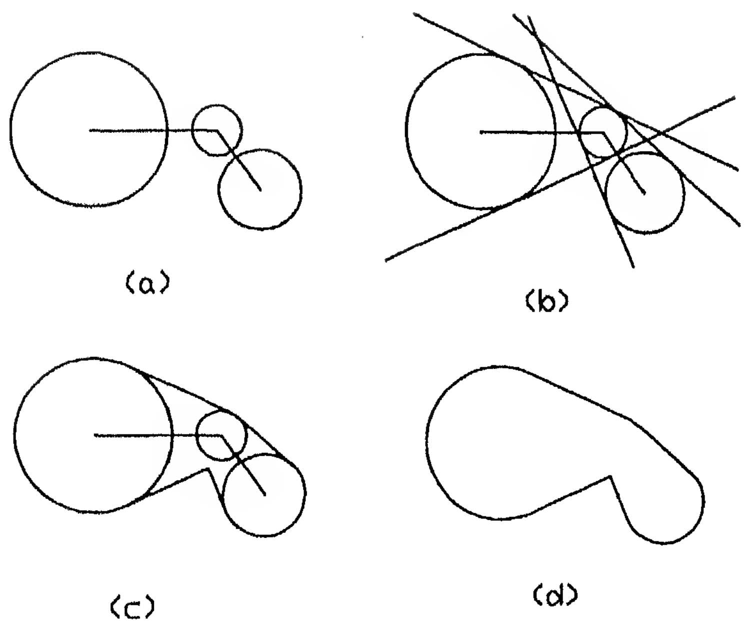

Figure 4.8: The reconstruction process. (a) The input MAT (b) Tangents on circles defined (c) Tangents trimmed (d) Final contour.

endcircles. Essentially, the process of reconstruction start from the root link and then recursively traverses to each child link. For a particular link, the tangents on the circles at end nodes are calculated. Thereafter, arcs and tangents are trimed to generate the enevelope of all tangents and circles, to define the boundary. The process of reconstruction is illustrated in fig. 4.8.

4.8 Fitness criteria for GA

Any MAT generated from the class of a given hybrid QMAT can be evaluated with respect to its *viability*, measured in GA with the *fitness* criterion, which can be defined according to particular application requirements.. The objective then becomes to select a shape which is most desirable (corresponding to maximum fitness or minimum penalty). Since GAs use a population of solution points, it is possible to deal with many shapes simultaneously. Every random MAT generated by GA is evaluated and given a suitable penalty value according to its degree of inconsistency / infeasibility and some other user-defined criteria. Our present implementation shows the results of a three-link case, where MAT consists of three links meeting at a point. The longest link has been taken as the reference length, and the length

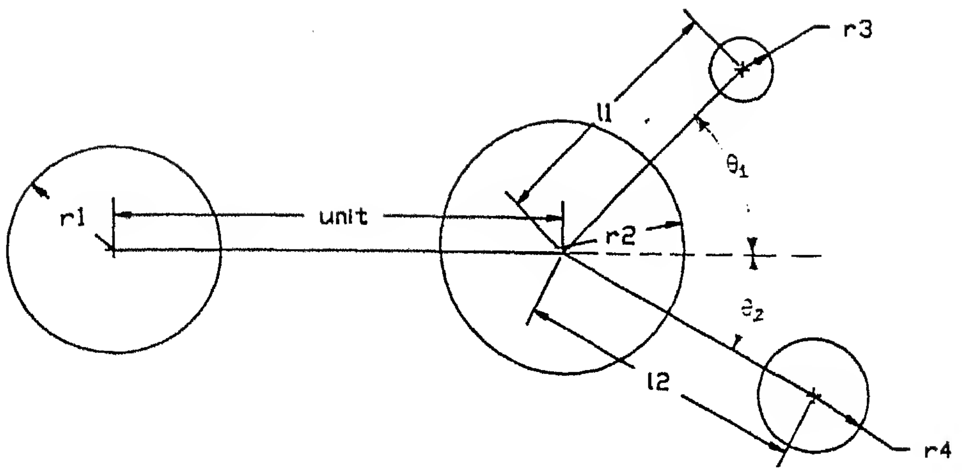

Figure 4.9: The basic geometry of a three-link MAT. The length of root has been taken as reference.

ratios of the other two are $\ell 1$ and $\ell 2$. Similarly the radius ratios at the four nodes are $r1$ through $r4$. The two angles made by $\ell 1$ and $\ell 2$ with the reference link are $\theta 1$ and $\theta 2$ (Figure 4.9). A qualitative version of this model will have all its parameter values specified in an uncertainity zone, which are shown as hatched areas in fig. 4.10. This figure also shows some of the shapes of the class represented by this model. Note that specifying other uncertainity zones on the parameter values of this model will result in a different class of shapes. For example, another class of shapes is shown in fig. 4.11 which relate to the rocker arm used in I.C. engines.

4.8.1 Penalties due to infeasibilities

As discussed in previous section, some MATs generated from the QMAT model may be infeasible. Such instances can be eliminated with a penalty function which increases with their degree of infeasibility. This penalty is usually of greater significance than penalties for contour properties. Minimization of summation of penalties in GAs guides the search to feasible shapes. We consider the two following cases as infeasible.

1. **Circle inclusion :** (Fig 4.7(a)). This is an infeasible case, because the tangents on the circles are undefined. Hence, it must be discarded by GA. For this purpose, we give a penalty value :

$$\phi = C_{inf}\left(1.0 + \left(\frac{(r_{max} - r_{min} - \ell)}{2r_{max}}\right)^2\right) \qquad \text{for } \ell \leq r_{max} - r_{min}$$

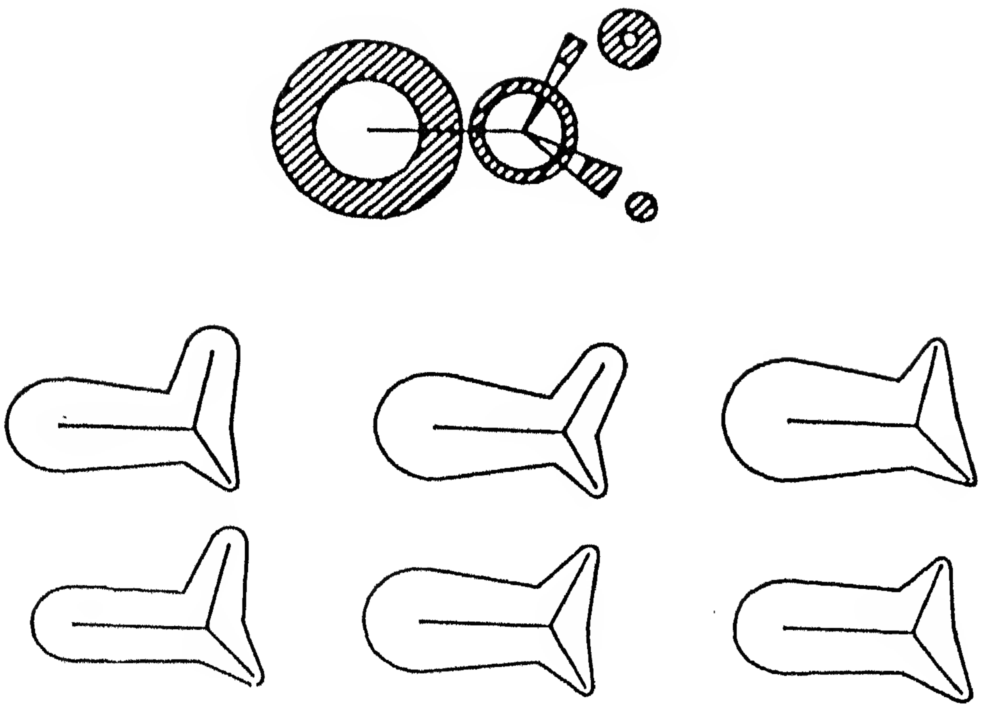

Figure 4.10: Qualitative version of three-link MAT, together with some of the shapes generated by this model. Note the abstract similarity between the shapes.

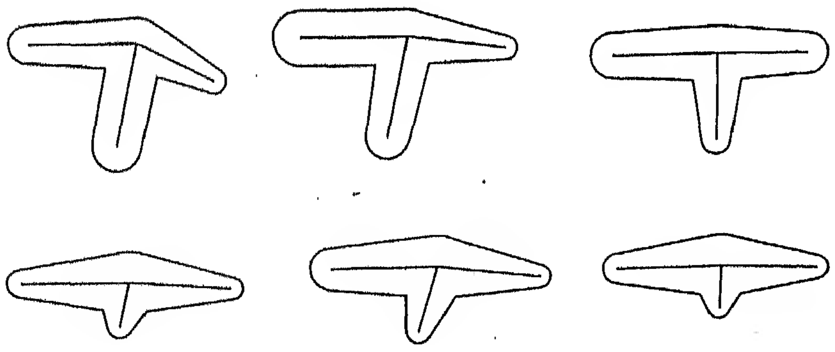

Figure 4.11: Some sample contours for an I.C. engine rocker arm. These were generated by the hybrid qualitative medial axis transform model.

where r_{max} and r_{min} refer to the bigger and smaller radius respectively and ℓ to the length of link. C_{inf} is the infeasibility coefficient setted by user (default value is 10.0). The above equation is chosen because it gives a non-zero monotonically increasing penalty with increasing infeasibility (Fig. 4.12 a).We can use any other function with a monotonically decreasing penalty as we move towards the feasibility zone.

2. **Boundary inclusion :** (Fig 4.7(b)). This is also an infeasible case, because the boundary of the contour is undefined at the inclusion point. This type of case is penalized by the following penalty function :

$$\phi = \sum_{i=1}^{2} \phi(i)$$

where $\phi(i)$ is calculated w.r.t. to i^{th} vertex of the boundary and the circle as below :

$$\phi(i) = \begin{cases} P_{cir} - C_{cir}\left(\frac{d_i}{r}\right)^2 & \text{if } d_i \leq r, \\ 0.0 & \text{otherwise.} \end{cases}$$

Here d_i is the distance of i^{th} vertex from the center of the circle and r is the radius of the circle. The coefficient P_{cir} and C_{cir} can be set by user (default values are 20.0 and 5.0). The penalty is a smooth function which increases as the infeasibility increases (Fig 4.12 b). Note that the penalty function increases as the included portion of boundary comes more near to the centre of circle. The penalty is equal to zero, if none of the portion of boundary is included in any of the circles.

4.8.2 Application-oriented penalties

In addition to the penalties discussed above, a number of other properties may be relevant depending upon the application. The user is given the flexibility to set different weights of these penalties, according to his requirements. Thus, if some weight is set to zero, the property corresponding to it gets effectively removed from consideration. This section illustrates some of the different contour properties that can be considered; note that the user can easily incorporate other functions of arbitrary complexity.

1. *Penetration of generating circles :* Too much penetration of the circles leads to near-circle shapes and small changes in the radii may even result in infeasible (Fig 4.5). The

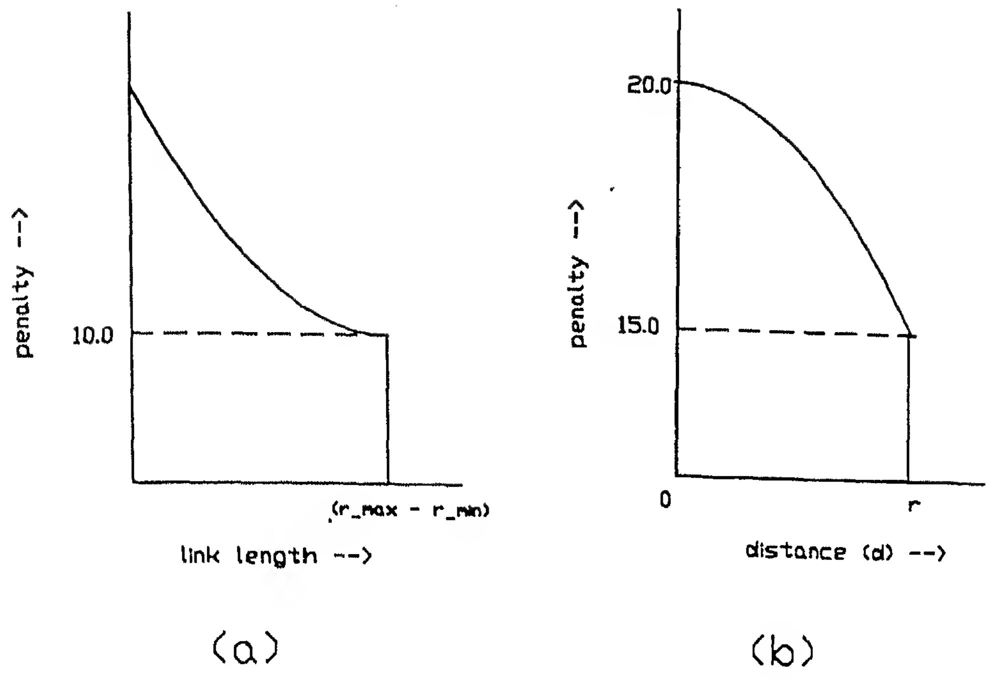

Figure 4.12: Penalties for infeasible MATs, (a) for circle inclusion, (b) for boundary inclusion.

penetration criterion is modelled as a penalty function that increases as two circles on the same link penetrate each other:

$$\phi = \left(\frac{(r_1 + r_2 - l)}{2r_{max}}\right)^2 \qquad \text{for} \quad \ell \leq r_1 + r_2$$

where r_{max} is the greater of (r_1, r_2), and l is the distance between the two centers (link length).

2. *Non-convexity :* In certain applications, non-convexity in the shape may be undesirable. The test function measures non-convexity in terms of reflex angles. If the reflex angle is more than π, we give a penalty equal to $(angle - \pi)^2$ else it is zero. This penalty is summed over all the reflex angles of the contour. Note that this is just one method of penalizing the non-convexity and there can be many possible methods to do this.

3. *Number of edges :* In general, a simple shape (containing lesser number of edges) may be preferred over its more complex counterparts. To model this criterion, we generate a penalty function equal to $k \times N$ where k is a normalization constant (we have used $k = 0.1$) and N is the actual number of edges in the contour.

4. *Area of the polygon :* This may need to be either minimized or maximized. Area can be computed based on the final contour obtained from the MAT instance, and the penalty

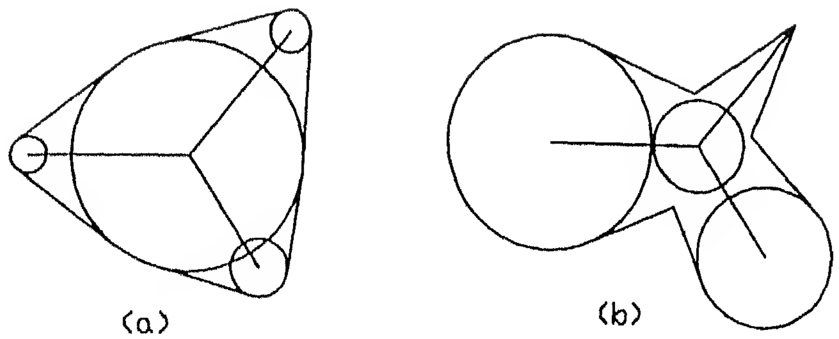

Figure 4.13: Examples of (a) smooth shape (b) non-smooth shape, generated by same MAT, but with different radii.

set at *weight* × *area* if minimizing (Weight is given a negative value if area is to be maximized).

5. *Smoothness :* Shapes with many sudden changes in the angle may be less desirable. Fig 4.13 shows examples of smooth and non-smooth shapes. To model this criterion, we traverse the whole contour, keeping track of change in the orientation from one edge to other. The penalty function is a summation of squares of these angle changes. Thus smooth shapes (with lesser change in angles) are given less penalty and sharp shapes are highly penalized.

6. *Aspect Ratio :* Often, as in the example of the knife in fig. 1.1 the aspect ratio of the shape (height/width) may be an important functional parameter required to fall within some prespecified range. The penalty function for this property first calculates the actual aspect ratio of the shape generated, and if the a.r. is out of the specified range, it sets a penalty value equal to square of deviation from the bounds, otherwise no penalty is given. The aspect ratio is calculated in the frame defined by the longest axis, i.e. the width along the longest axis link, and height along a perpendicular direction.

7. *Parallelism of edges :* In some applications (especially engineering applications and other artifacts) some edges may be exactly parallel. If this penalty function is invoked, it checks the angle between the orientation of two tangents of a link. Then the penalty is calculated as follows where θ is the angle between the two edges:

$$\phi(i) = \begin{cases} C_p\,\theta & \text{if} \quad 0.0 \le \theta \le \alpha_1, \\ C_p\alpha_1 & \text{if} \quad \alpha_1 \le \theta \le \alpha_2, \\ 0.0 & \text{otherwise.} \end{cases}$$

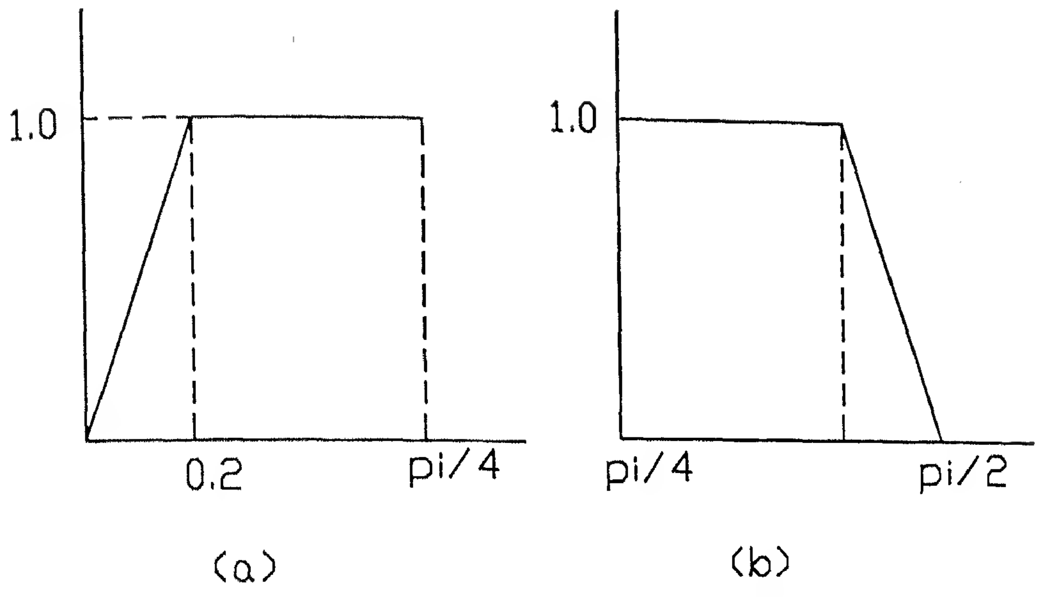

Figure 4.14: Penalty function for (a) parallelism (b) perpendicularity. x-axis on both figures represents the angle θ between two edges being compared.

The coefficient C_p used in this and following equation is a slope factor (we have taken it as 5.0). Angle constants α_1 and α_2 are some predefined threshold values (we have used 0.2 and $\pi/4$ respectively). This penalty is as shown in fig. 4.14(a). The effect of this penalty is that there is no discrimination among shapes having $\theta > \alpha_1$ as far as this penalty is concerned. This is done because if the angle between the tangents is more than the threshold value, they can't be termed as "nearly" parallel, and are therefore, equally good.

8. *Perpendicularity of edges* : Similar to previous one, some applications may need an exact perpendicularity between two neighbouring edges. Calculation of this penalty is similar to the previous one, except that the ranges for θ have been changed.

$$\phi(i) = \begin{cases} C_p(\pi/2 - \theta) & \text{if} \quad \pi/2 - \alpha_1 \le \theta \le \pi/2, \\ C_p\alpha_1 & \text{if} \quad \pi/2 - \alpha_2 \le \theta \le \pi/2 - \alpha_1, \\ 0.0 & \text{otherwise.} \end{cases}$$

This penalty is as shown in fig. 4.14(b). Similar to previous one, this penalty function treats all those cases same which have angle θ less than threshold value, since all such cases fall out side the definition of "nearly" perpendicular.

In the reconstruction process of a given QMAT, some axis segments may have been deleted, resulting in uncertainity about the edge profiles there (see fig. 4.4(a)). In the GA, this can be

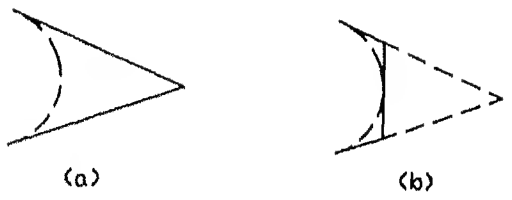

Figure 4.15: Linearizing an arc (a) with extended tangents, (b) with truncated tangents.

adjusted by the user, who can specify how the arcs of the contour will be displayed. These arcs may emerge because of removal of some links, and/or having a non-zero radius for an end node. The arcs may remain arcs or they can be converted into line segments depending upon following two parameters :

1. Extended-tangent linearity : If this is set equal to 1.0, all the arcs having sweep angle less than $\Theta_e(= \pi/2)$ are converted into two extended tangents. This is shown in fig 4.15(a).
2. Truncated-tangent linearity : If this is set equal to 1.0, all the arcs having sweep angle less than $\Theta_{th}(= \pi)$ but greater than $\Theta_{tl}(= \pi/2)$ are converted into three line segments, the middle one being the tangent at mid point of the arc. This is shown in fig 4.15(b).

If these degrees are set to 0.0, all arcs remain arcs. Any value of these parameters in the range (0,1) leads to some of the arcs being converted to lines depending upon their sweep angles. It results in a more angular look to the shape.

In some sense, non-convexity in the contour dictates inaccuracy in the input MAT, because MAT of such a contour will have, in general, some portion of parabola. The present implementation consists of only straight lines. We are allowing this approximation to make the implementation simple.

4.9 Results

As discussed earlier, we have used a GA approach to find a desirable shape from QMAT model. The objective function for minimization is the summed penalty functions discussed in previous section. We have used GA with SBX crossover operator with binary tournament selection. For testing purpose, we have defined some test functions according to previous section and generated some shapes. We have used following parameter values which were kept constant for all runs[1].

1. Population Size : 50
2. Cross over probability : 1.0
3. Mutation probability : 0.01
4. Max. number of generations : 40
5. Exponent(n) : 2.0 (Fixed throughout)

In the following test cases the user is assumed to have provided ranges of variability associated with each parameter in the model. Thus, by fixing the angle between two links but by allowing some variation in the radii, he/she can emphasize the functional relevance of the angle while stating that the actual shape and other dimensions are not as important. Furthermore, the user can specify the optimization criteria to be used by the GA (by setting appropriate weights to each penalty functions) and after viewing the results, he/she may change either the original input or the fitness criteria. To illustrate this procedure, we will trace the development of several themes – a fish like shape, an L-shaped bracket and a long forking building. In each of these instances, a set of QMAT data is provided by the user (Table 4.1), and we show the results obtained by adjusting the fitness criteria. In each instance, the numbers in parenthesis represent the range of values possible in the HQMAT at the desired level of inexactness.

Table 4.1: QMAT data used for example shapes

Variable	Fish	Bracket	Building
$r1$	(0.5,0.8)	(0.0,0.5)	(0.1,0.3)
$r2$	(0.0,0.3)	(0.2,0.2)	(0.1,0.3)
$r3$	(0.0;0.3)	(0.0,0.5)	(0.1,0.3)
$r4$	(0.0,0.3)	(0.0,0.0)	(0.1,0.3)
$\ell 1$	(0.7,0.7)	(0.3,1.0)	(0.3,1.0)
$\ell 2$	(0.7,0.7)	(0.1,0.5)	(0.3,1.0)
$\theta 1$	(45,90)	(90,90)	(45,45)
$\theta 2$	(-90,-45)	(-45,-45)	(-45,-45)
Fitness Criterion	Non-convexity smoothness	parallelism perpendicularism aspect ratio	parallelism perpendicularism aspect ratio area

[1] However, it may be a good idea to change these parameter values and investigate its effects on performance of GA runs in this particular application.

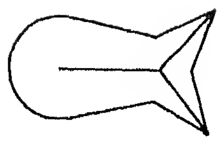

Figure 4.16: The final fish-shape obtained by GA.

4.9.1 Biological shape (fish)

Here the user is interested in a bulbous shape for the head area and a not-too-angular tail. Smoothness in the final shape is desirable since biological shapes are smooth. Also, non-convexity has been used here to reduce sharp angles in the tail (non-convexity increase with the reflex angles). The head radius is significantly larger than the other radii, and linearity is not desirable since the shape is to remain rounded. A class of fish shapes generated by these values were shown in fig. 4.10, while the final version of shape is as depicted in fig. 4.16.

4.9.2 Bracket cross-Section design

Here we are interested in designing a bracket to support a shelf, say. The shape of it is roughly like an inverted "L". Fig. 4.17 shows some possible shapes which are obtained by using the parallelism and perpendicularism constraints and the aspect ratio (how wide is the shelf vs. how much wall support is desirable).

One point to be noted here is that in a conceptual design system dealing with such objects the user may be allowed to add other functional constraints, such as the strength constraint representing the load to be carried by the wall and the strength of the walls to support it. These functional specifications can be automatically optimized along with the shape in obtaining valid designs (Mukerjee and Srivastava, 1994).

4.9.3 Building design

Consider the ground plan of a building that is to have a long central part, with two branching corridors at one end. In this example, the relative lengths of the corridors are controlled by the aspect ratio parameter, and the widths by the area parameter (fig. 4.18). All test cases

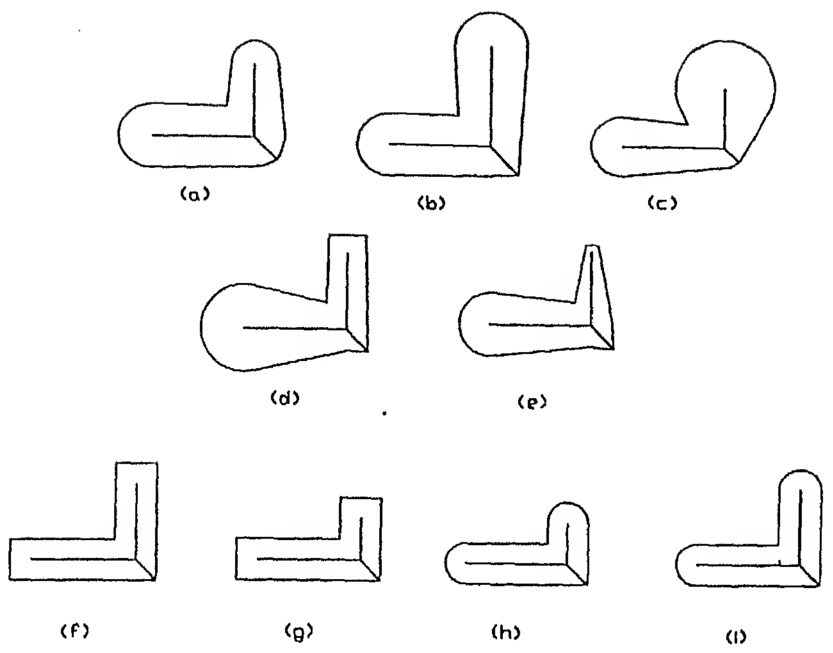

Figure 4.17: The bracket cross-section is an L. (a) ... (e) shapes from early generations of GA run. (f) final shape with aspect ratio = 0.8 (g) shape with a.r. = 0.5 (h) shape with a.r. = 0.5 (linearity turned off) (i) shape with a.r. = 0.8 (linearity turned off).

use parallelism and perpendicularism. The degree of linearity is kept 1.0 to allow linearized shapes.

4.9.4 Some more test cases

In order to bolster further our claims, we have generated few more test cases. Fig. 4.19 shows a test case where area minimization alone was used as GA fitness criterion. The same input model was used to generate another test case with area maximization as the fitness criterion. Further, we claim that our three-link MAT model itself is able to represent a wide variety of shapes. Some of these can be seen in fig. 4.21. However, it is worth mentioning here that a general shape may have an MAT consisting of more than three-link. Qualitative models for a generalized case can be developed and tested along same lines.

4.10 Summary

In this work we have illustrated an innovative approach for representing ambiguous or ill-defined geometries. The model shown here uses as input an axial model of the shape. The basic technique has a number of possible applications — conceptual design visualization, models for

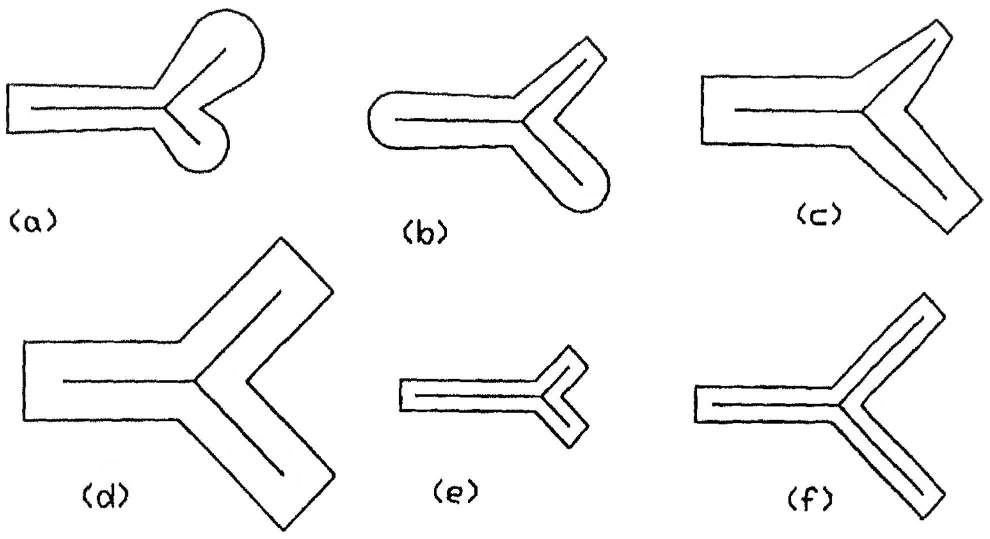

Figure 4.18: Shapes for forking building. (a) ...(c) Shapes from early generations of GA run. (d) final shape for aspect ratio = 1.0 together with area maximization constraint. (e) shape for a.r. = 0.5 with area minimization (f) shape for a.r. = 1.0 with area minimization.

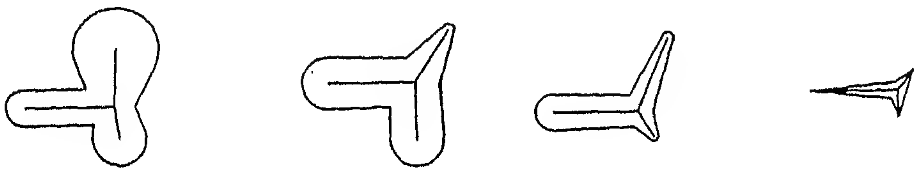

Figure 4.19: A test case for area minimization. First three figures are early samples of GA run. The last shape is optimized. Note that the user has defined the uncertainity zones in a manner that no better shape (with lesser area) is possible.

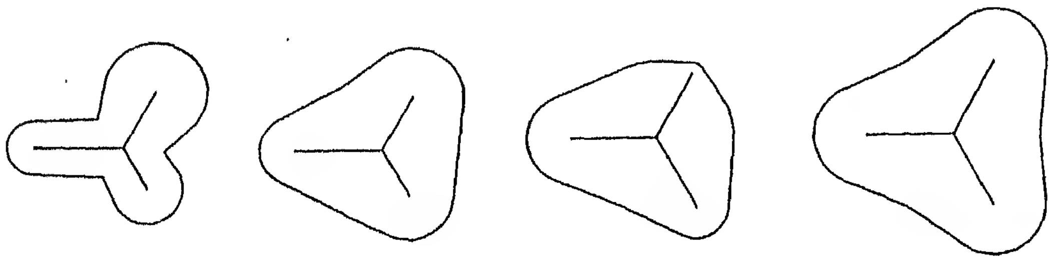

Figure 4.20: A test case for area maximization. First three figures are early samples of GA run. The last shape is optimized. Note that the user has defined the uncertainity zones in a manner that no better shape (with more area) is possible.

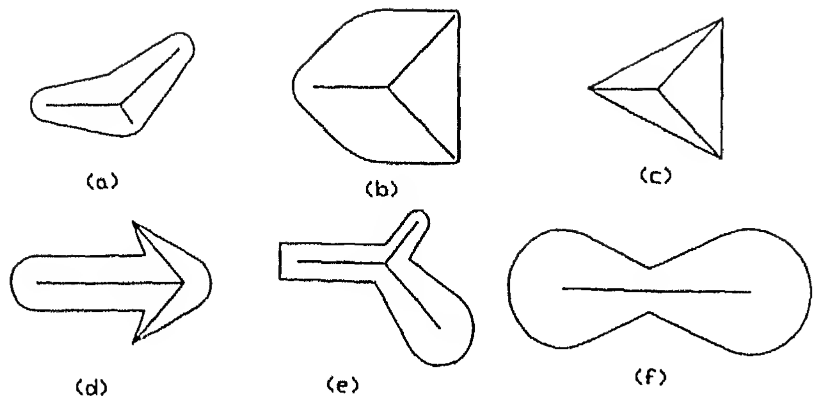

Figure 4.21: A variety of shapes can be represented by three-link MAT. (a) An arm shape (b) A bracket shape (c) A triangle (d) An arrow (e) A pistol shape (f) A dumbell shape.

computer vision, and any other application involving sketch work or where shapes are to be visualized. The genetic algorithms approach used here provides a mechanism for searching non-monotonically in an extremely nonlinear domain. Also, by being able to visualize samples from all generations, the system also provides the user with some idea of the variations possible on the basic shape constraints.

Chapter 5

Extensions

The study made in this thesis opens a number of ways for possible extensions. We discuss the extensions of GA part and MAT part separately.

5.1 Extensions (real-coded GAs)

1. *Other techniques for bounded search space*

   In the present study, we have either allowed the children points to lie anywhere between negative infinity to positive infinity, or have restricted them to remain within bounded search space using scaling technique (see section 2.4.3 in chapter 2). However, This scaling creates an artificial bias of the children points towards middle of the search space. It would be interesting to investigate this phenomena in more detail in binary GA, and then simulate that in SBX.

2. *Study of effect of exponent n on GA performance*

   In this thesis, we have chosen some values of exponent n and reported the results of GA performance. However, we feel that this is an important parameter in terms of search power of crossover. A low value of n signifies broad search and a high value a local search. It may be interesting to carry out a parametric study on GA performance with respect to n. This exponent can also be varied during the GA search, namely a low value in early generations, and a high value for later generations.

3. *Selection of other form of probability distributions for contracting and expanding crossovers*

   The probability distribution for contracting crossover used in this thesis is a polynomial function to the spread factor. This distribution is used to simplify the computations required in each crossover operation. Other functional forms of the probability distributions can also be used, but care must be taken to choose a distribution which is similar in form to that given in figure 2.6. In this respect, a Gaussian-like distribution with mean at $\beta = 1$ and variance (σ^2) chosen to have at most $\pm 3\sigma$ points within a specified bound can also be used.

4. *Real-coded mutation operator*

   In this thesis, our main thrust was on investigation of the effect of the crossover operator mainly. Mutation operator was developed (see section 2.5) in chapter 2 but its effects are not investigated. A future study may incorporate investigation of effect of mutation and crossover both used together. Moreover, there can be other forms of mutation operator than that used in this thesis. However, any such form should have the property that near points should be more probable to be generated than far-off points.

5. *Application of SBX to real-world optimization problems*

   The present study shows that real-coded GAs are preferred to binary-coded GAs for problems with continuous search space and for problems where arbitrary precision is desired. For example, we have used real-coded GAs with SBX in a robotics path planning problem (Mukerjee, Sharma, and Agrawal, 1995). Many engineering design optimization problems contain mixed variables—both discrete and continuous. The classical optimization methods do not perform very well in solving these kinds of problems (known as mixed integer programming problems). GAs with a combination of both binary coding (for discrete variables) and real coding (for continuous variables) can be used to solve these problems efficiently. For example, SBX is currently being used in optimization of a structural truss (Chaturvedi, 1995). Traditionally, a truss is optimized in two phases. First, a fixed values of areas for its members is assumed and then its topology is optimized. After suitable topologies have been found, optimization techniques are used to decide the areas of its individual members. But the optimization in two stages does not guarantee to lead to global optimal solution. Hence topology and areas should be optimized simultaneously. Thus, it becomes a mixed variable problem, because areas are continuous variables and topology can be represented by discrete variables. SBX with single point binary crossover is being used for this problem. The crossover operation is performed as follows. A variable is first chosen for crossover. If a discrete variable needs to be crossed, the single-point crossover is used to the bits representing that variable only. But if a continuous variable needs to be crossed, the SBX operator is used. This way the resulting string also becomes a valid string.

6. *Simulating other binary crossovers for real-coded GAs*

   The SBX operator developed here is based on the probability distribution of children strings in a single-point crossover. Similar probability distributions (a function of the spread factor β) can also be obtained for multi-point and uniform crossovers and other similar real-coded crossovers can be designed. Probability distributions similar to the one developed here are expected, it may be interesting to investigate whether they turn out otherwise. Depending on the outcome of that analysis, the issue of superiority of one crossover over other can be resolved, particularly in the context of GA's application to continuous variables.

7. *Population sizing and convergence proof*

   Throughout in this study, reasonable values of the population size are used. As demonstrated elsewhere (Goldberg, Deb, and Clark, 1992), the proper population sizing is important for successful working of GAs. That study also suggested a population sizing based on the ratio of the difference in detectable function values to the variance of the function. An attempt can be made to derive a population sizing for real-coded GAs from a similar viewpoint. In binary-coded GAs, the major difficulty towards achieving a convergence proof lies in the mechanics of the crossover operator. Although the mechanics of the single-point crossover (or for that matter any other binary crossover operator) is simple, their mathematical modeling becomes difficult. Any convergence proof of a stochastic process involves calculation of transition probabilities. Since in the SBX operator, a direct probability distribution of children points for any given pair of parent points is used, the proof of convergence of genetic algorithms may become easier.

8. *Multimodal and multiobjective optimization using real-coded GAs*

   The success of binary GAs in many single-objective, unimodal problems has led researchers to extend GA's applicability to other types of optimization problems—multimodal and multiobjective function optimization problems. In order to find multiple optimal solutions simultaneously, *sharing* functions are used to create artificial fitness values, which are used in the reproduction operator (Deb, 1989; Goldberg and Richardson, 1987). In order to find multiple pareto optimal points for multiobjective function optimization, the concept of *nondomination* is used to create artificial fitness values, which are again used in the reproduction operator (Srinivas and Deb, in press). Since only reproduction operator needs modification to solve the above problems, similar reproduction techniques can be used along with the SBX operator used in this study to investigate the efficacy of real-coded GAs in multimodal and multiobjective problems defined on a continuous search space.

5.2 Extensions (ambiguous shape modelling)

1. *Model for n-link case*

   The present implementation uses MAT composed of 3-link as the basic input. In real-world applications, shapes may be much more complicated involving large number of links in their MATs. Thus, a generalization of the present implementation is needed where model should be able to handle n-link MAT. It is suggested that the routines developed in the present implementation can be used as basic computation systems to be applied at each node of n-link MAT.

2. *MAT model from a 'sketch' of the shape*

   Right now, the user has to input the model of QMAT specifying the uncertainities for each parameter. Even though this was satisfactory for some test cases as shown in the thesis, it will be inconvenient for real-world applications involving complicated shapes Thus, it is suggested that the MAT information can be obtained from an input 'sketch', using a MAT-calculation system based on any standard algorithm for creating voronoi diagram of line sites (Fortune, 1987; Tiwari, 1986; Yap, 1987). Thus the user won't need to even think of MAT at all. Instead of that, he will specify some initial rough sketch of shape, specifying his degree of uncertainity. Based on this input, the model should be able to find out the MAT and its qualitative version, and feed it to the main program to generate a set of qualitative similar shapes.

3. *Enhanced user interface*

   A good user-interface is lacking in this work. The main GA program runs on UNIX based systems and the input data has to be given through keyboard/file. The graphic display program runs on MS-DOS platform, and uses the datafiles generated by the main program. The future implementation should integrate all the subsystems of the model on a single plateform. The user must be able to see various possible alternative shapes dynamically. Then he/she can specify (possibly using a mouse) which one is most suitable for his application and the program should now take this shape as the input for next iteration.

4. *Other fitness criteria*

In the present implementation, we have kept a limited number of fitness criteria for the sake of simplicity. There are, of course, endless number of contour properties that could have been tackled by GA, for example convexity ratio (area of polygon/ area of it's convex hull), first and second moment of inertias, perimeter distance, noise (small sharp-cornered edges) and other similar issues. Moreover, in the real-world applications, the list of fitness criterion may be much more than this one, because it will include some application-oriented criterion also, eg. strength of the part under certain load conditions, manufacturability consideration etc. If future implementation is carried out in C^{++} under OOPS (Object oriented programming system), the extension to the list of fitness criterion may be easier.

5. *Empirical testing of GA parameters*

In the test results presented here, we have kept a fixed value for each GA parameter. However, it may be a good idea to vary these parameters and investigate their effect on this particular application. It may be found that a particular set of input data may be best handled by a particular set of GA parameters. For example, problems involving large variation in MAT parameters tend to become more complex, and may require larger population sizes. A high mutation probability may be suitable in the applications where user is more interested in diversity of the shapes rather than it's optimization. A research on these issues may provide some empirical guidelines to user about what GA parameters to use for a particular application.

Chapter 6

Summary and conclusions

In this thesis, a real-coded crossover operator has been developed based on the search characteristics of a single point crossover used in binary-coded GAs. In order to define the search power of a crossover operator, a spread factor has been defined as the ratio of the absolute differences of the children points to that of the parent points. Thereafter, the probability of creating a child point for two given parent points has been derived for the single-point crossover. Motivated by the success of binary-coded GAs in problems with discrete search space, simulated binary crossover (SBX) operator has been developed to solve problems having continuous search space. The SBX operator has similar search power as that of the single-point crossover.

On a number of test functions including De Jong's five test functions, it has been found that real-coded GAs with the SBX operator perform as good or better than binary-coded GAs with the single-point crossover. In comparison with another real-coded crossover operator (BLX-0.5) suggested elsewhere, SBX performs better in difficult test functions. It has been observed that SBX is particularly useful in problems where the bounds of the optimum point is not known a priori and where there are multiple optima, of which one is global.

Real-coded GAs with the SBX operator have been also used to solve a blocked function, introduced by Goldberg (1991). Blocked functions are difficult for real-coded GAs, because local optimal points block the progress of search to continue towards the global optimal point. An active, two-sided blocked function has been designed and solved using GAs with the SBX operator. The simulation results have shown that in most occasions, the search proceeds the way as predicted, but SBX can overcome the blocking by the local optimal points and converge to the global optimal solution.

These results are encouraging and suggest avenues for further research. Because the SBX operator uses a probability distribution for choosing a child point, the real-coded GAs with

SBX are one step ahead of the binary-coded GAs in terms of achieving a convergence proof for GAs. With a direct probabilistic relationship between children and parent points used in this thesis, cues from the classical stochastic optimization methods can be borrowed to achieve a convergence proof of GAs, or a much closer tie between the classical optimization methods and genetic algorithms is on the horizon.

In the second part of the thesis, attention has been focused on a real-world application for GA with SBX. Here, a less-precise model (a hybrid qualitative model) is used to represent a large class of possible shapes. GA with SBX has been used to find out desirable shapes from the class of shapes represented by QMAT, and to remove infeasible or undesirable ones from consideration. The system is designed in such a way that a user can identify his application oriented desirability criteria and set the weight to each criterion accordingly. Several sample shapes developed by this system have been shown, which reveal the usefulness of this system in conceptual design stage. Although this is intended to be an application test for GA, the problem is of sufficient importance of its own in the areas of shape approximation and conceptual design.

References

Agrawal, R. B., Mukerjee, A. and Deb K. (1995). Modelling of inexact 2-D shapes using real-coded genetic algorithms, In P.K. Roy and S.D. Mehta (Eds.) *Proc. of Symposium on Genetic Algorithms*, pp 41–49.

Back, T., Hoffmeister, F., and Schwefel, H. P. (1991). A survey of evolution strategies. In R. K. Belew and L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 2–9).

Blum, H. (1967). A transformation for extracting new descriptors of shape, In W. Dunn (Ed.), *Perception of Speech and Visual Form*, The MIT Press, Cambridge, MA, pp 362–380.

Booker, L. B. (1993). Recombination distribution for GAs. In D. Whitley (Ed.), *Foundations of Genetic Algorithms* (pp. 29–44).

Chaturvedi, D. (1995). Structural optimization using real-coded GAs. Master's thesis, Dept. of Civil Engineering, Indian Institute of Technology, Kanpur.

Davis, L. (1991a) Hybridization and numerical representation, In L. Davis (Ed.) *The Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, pp 61–71.

Davis, L. (1991b). Bit-climbing, representational bias, and test suite design. In R. K. Belew, & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, (pp. 18–23).

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems (Doctoral dissertation, University of Michigan, Ann Arbor). *Dissertation Abstracts International, 36*(10), 5140B. (University Microfilms No. 76-9381)

Deb, K. (1989). Genetic algorithms in multimodal function optimization, *Master's Thesis*, (TCGA Report No. 89002). Tuscaloosa: University of Alabama.

Deb, K. (1993). Genetic algorithms for engineering design optimization. In J.N. Reddy, C.S. Krishnamoorthy, and K.N. Seetharamu (Eds.) *Advanced Study Institute on Computational Methods for Engineering Analysis and Design,* (pp 12.1–12.25).

Deb, K. and Agrawal, R. B. (1994). Simulated binary crossover for continuous search space, TR No. ME/94027, Dept. of Mechanical Engg., IIT Kanpur 208016, India.

Eshelman, L. J. and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval schemata. In D. Whitley (Ed.), *Foundations of Genetic Algorithms, II* (pp. 187–202).

Fortune, S. (1987). A sweepline algorithm for voronoi diagrams, *Algorithmica*, vol.2. pp. 153–174.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading: Addison-Wesley. pages 412.

Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems, 5*(2), 139–168. (Also IlliGAL Report 90001)

Goldberg, D. E., and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.) *Foundations of Genetic Algorithms,* pp. 69–93.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems, 6*, pp. 333–362.

Goldberg, D. E., and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms,* pp. 41–49.

Gordon, V. S. and Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms,* pp. 177–183.

Holland, J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, The University of Michigan Press.

King, J. S. (1991). Inexact visualization : qualitative shape representation for recognizable reconstruction, Master's thesis, Dept. of Comp. Sc., Texas A & M Univ.

King, J. S., and Mukerjee, A. (1990). Inexact Visualization, In *Proceedings of the IEEE Conference on Biomedical Visualization,* Atlanta GA, pp. 136–143.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. In *Science*, vol. 220, Number 4598, pp. 671–680.

McCullough, M. (1990). Computer aided conceptual design ? In *CADENCE,* July 1990, pp. 31–34 and 123.

Mittal, N., and Mukerjee, A. (1995). Qualitative subdivision algebra: Moving towards the quantitative, submitted to IJCAI-95. Also available as a TR, Dept. of Mechanical Engg., IIT Kanpur.

Mukerjee, A. and Joe, G. (1990). A qualitative model for space. In 8th-AAAI (1990), pp. 721–727. (Longer version in Texas A&M University, Computer Science Dept, TR 92-003, 70 pages).

Mukerjee, A., Sharma, S. P. and Agrawal, R. B. (1995). When is an obstacle a *perfect* obstacle ? *Proceedings of the IEEE Conference on Robotics and Automation,* Nagoya, Japan.

Mukerjee, A. and Srivastava S., (1994). Designing a class of geometries — from concept to instantiation, TR No. ME/94025, Dept. Mechanical Engg., IIT Kanpur 208016, India.

Muhlenbein, H., Schomisch, M., and Born, J. (1991). The parallel genetic algorithms as function optimizer. In R. K. Belew and L. B. Booker (Eds.) *Proceedings of the Fourth International Conference on Genetic Algorithms,* (pp. 271–278).

Peters, J., Leuven, K. U., Krause K. L., Berlin, T. U., Agerman, E. and Stockholm, K. T. H. (1990). Design : an integrated approach. *Annals of the CIRP,* vol. 39/2/1990, pp. 599–607.

Radcliffe, N. J. (1993). Genetic set recombination. In D. Whitley (Ed.) *Foundations of Genetic Algorithms II,* pp. 203–219.

Rao, S. S. (1991). *Optimization theory and application.* New Delhi, Wiley Eastern Ltd.

Requicha, A. A. G. (1980) Representation for rigid solids: Theory, methods, and systems, In *ACM Computer Surveys*, Dec. 1980.

Sandor, G.N. (1964). The seven stages of engineering design. In *Mechanical Engineering*, April 1964, pp. 21–25.

Schaffer, J. D., Caruana, R. A., Eshelman L. J. and Das R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization, In Schaffer, J. D. (Ed.) *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 51–60.

Spears, W. M. and De Jong K. A. (1991). An analysis of multi-point crossover, In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms*, pp. 301–315.

Srinivas, N. and Deb, K. (in press). Multiobjective function optimization using nondominated sorting genetic algorithms, *Evolutionary Computation*, MIT Press.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, (pp. 2–9).

Tiwari, A. (1986). Understanding biological form : A knowledge based environment to study form change, Master's thesis, Dept. of Comp. Sc., Texas A & M Univ., pages 219.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms*. (pp.205–218).

Yap, C. K. (1987). An *O*(*n log n*) algorithm for the voronoi diagram of a set of simple curve segments, In *Discrete Computational Geometry*, vol. 2, pp. 365–394.

APPENDIX–I

A program 'rga.c' was written in 'C' programming language to test our SBX crossover approach in this thesis. This is a GA implementation and have following features :

- *Variable representation :* Mixed (binary + real-coded). Can be used as purely binary GA or purely real-coded GA depending on user's wish.
- *Selection scheme :* tournament selection (size of tournament can be fixed by user)
- *Crossover :* For binary strings — single point crossover
  For real numbers — SBX / BLX-0.5
- *Crossover approach for multi-variable problems :* All three approaches of section 2.4.4
- *Mutation :* For binary strings — bitwise mutation
  For real numbes – real-coded mutation of section 2.5

In addition to this, the user can set following toggle switches :

- Variable bounds : fixed / flexible
- Evaluation : with sharing / without sharing
- Report : end report / generation-wise report

One particular GA run is terminated if any one of the following criteria is satisfied :

1. Premature convergence.
2. Target fitness value achieved.
3. Global solution is achieved.
4. Maximum number of generations over.
5. Population has spread in a very wide range (specified by user).

Since this program was intended to be used for testing purpose, it has the option of running the GA program many times (user can specify the number of runs) with different seed numbers generated automatically in the program. A final report is prepared in the file 'realga.out' for inspection. Following is a sample output file generated by the program.

```
=============================================
               INITIAL REPORT
=============================================
 CROSSOVER TYPE :  Binary GA (Single-pt);  Real GA (SBX)
 STRATEGY : Uniformly all variables 50
 TYPE OF BOUNDARIES :  Rigid
 Population size              : 50
 Total no. of generations     : 40
 Cross over probability       : 1.0000
 Mutation probability         : 0.0100
```

```
Sharing not to be done :
String length              : 15
Number of variables
               Continuous  : 2
               Discrete    : 1
                 TOTAL     : 3
Total Runs to be performed : 4
Epsilon for convergence    : 0.001
Epsilon for closeness      : 0.001
Distribution index         : 2
Tournament size            : 2
Target Objective fn. value : 0
Lower and Upper bounds     :
   -1.0000   <=   x1   <=   1.0000
   -1.0000   <=   x2   <=   1.0000
   -1.0000   <=   x3   <=   1.0000
=================================================

Run No.   Best X    Trials          Spread              Comment/Best String
=================================================================
  1.    0.0008850    1300    -0.935789 to   0.563219
       -0.0009773            -0.009467 to   0.172813
       -0.0002669            -0.010862 to   0.006025  100000000001110
          OBJECTIVE =  0.0000018
       Random Seed Number : 0.4000                  -> Target found
  2.   -0.0008850    1750    -0.250771 to  -0.000031
        0.0004291            -0.006790 to   0.006809
        0.0009418            -0.004047 to   0.008719  011111111110001
          OBJECTIVE =  0.0000019
       Random Seed Number : 0.5200                  -> Target found
  3.    0.0009461    1800    -0.980895 to   0.750420
        0.0003447            -0.006470 to   0.003524
       -0.0006636            -0.005220 to   0.001924  100000000001111
          OBJECTIVE =  0.0000015
       Random Seed Number : 0.6400                  -> Target found
  4.   -0.0001526    2050    -0.515732 to   0.999817
       -0.0006892            -0.001662 to   0.002052
       -0.0010936             0.001370 to   0.002423  011111111111101
          OBJECTIVE =  0.0000017
       Random Seed Number : 0.7600                  -> Not Found
=================================================================
 Average trials given =   1616.7
 Population prematurely converged 0 times
 Total Successes = 3 out of 4
 Average of best objective = 0.000002
=================================================================
```

ME-1995-M-AGA-SIM